

PHP快速入门及项目实战

快速服务：微博、QQ在线服务

自学视频：72集大型多媒体自学视频

海量资源：模块库、案例库、素材库、题库

潘凯华　李　慧　刘　欣　等编著

本书提供了内容丰富的配套资源，可以登录www.tup.com.cn，找到本书后，在该页面的“网络资源”超链接处下载。也可以访问本书的新浪微博，根据提示链接下载。

清华大学出版社

PHP快速入门及项目实战

潘凯华　李　慧　刘　欣　等编著

（72集大型多媒体自学视频）

（模块库、案例库、素材库、题库）

（微博、QQ、论坛技术支持）

清华大学出版社

北　京

内 容 简 介

本书系统、全面地讲解了使用PHP语言进行编程的各种技术，全书分为4篇18章，其中第1篇为基础篇（第1～8章），主要内容包括：PHP概述，PHP基础，PHP函数，PHP流程控制语句，PHP数组，Web技术，MySQL数据库和PHP数据库编程技术；第2篇为技能提高篇（第9～13章），主要内容包括字符串高级处理，日期和时间处理，图形图像处理，文件目录处理和面向对象编程；第3篇为高级应用篇（第14～17章），主要内容包括PDO数据库抽象层，Smarty模板，ThinkPHP框架和PHP的字符编码；第4篇为实战项目篇（第18章），通过模拟hao123网站，开发明日导航网站，以此来巩固所学基础知识，掌握应用ThinkPHP框架开发Web项目的精髓。

本书提供了大量的自学视频、源程序、素材，提供了相关的模块库、案例库、素材库、题库等多种形式的辅助学习资料，还提供迅速及时的微博、QQ、论坛等技术支持。

本书内容详尽，实例丰富，非常适合作为零基础学习人员的学习用书和大中专院校师生的学习教材，也适合作为相关培训机构的师生和软件开发人员的参考资料。

图书在版编目（CIP）数据

PHP快速入门及项目实战/潘凯华，李慧，刘欣等编著．—北京：清华大学出版社，2012.2

ISBN 978-7-302-27307-3

I．①P…　II．①潘…　②李…　③刘…　III．①PHP语言-程序设计　IV．①TP312

中国版本图书馆CIP数据核字（2011）第233242号

责任编辑：赵洛育　刘利民
版式设计：文森时代
责任校对：张彩凤
责任印制：
出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦A座　　**邮　　编：**100084
　　社 总 机：010-62770175　　**邮　　购：**010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　　张：**27.5　**字　　数：**794千字
版　　次：2012年2月第1版　**印　　次：**2012年2月第1次印刷
印　　数：1～5000
定　　价：49.80元

产品编号：043927-01

前　言

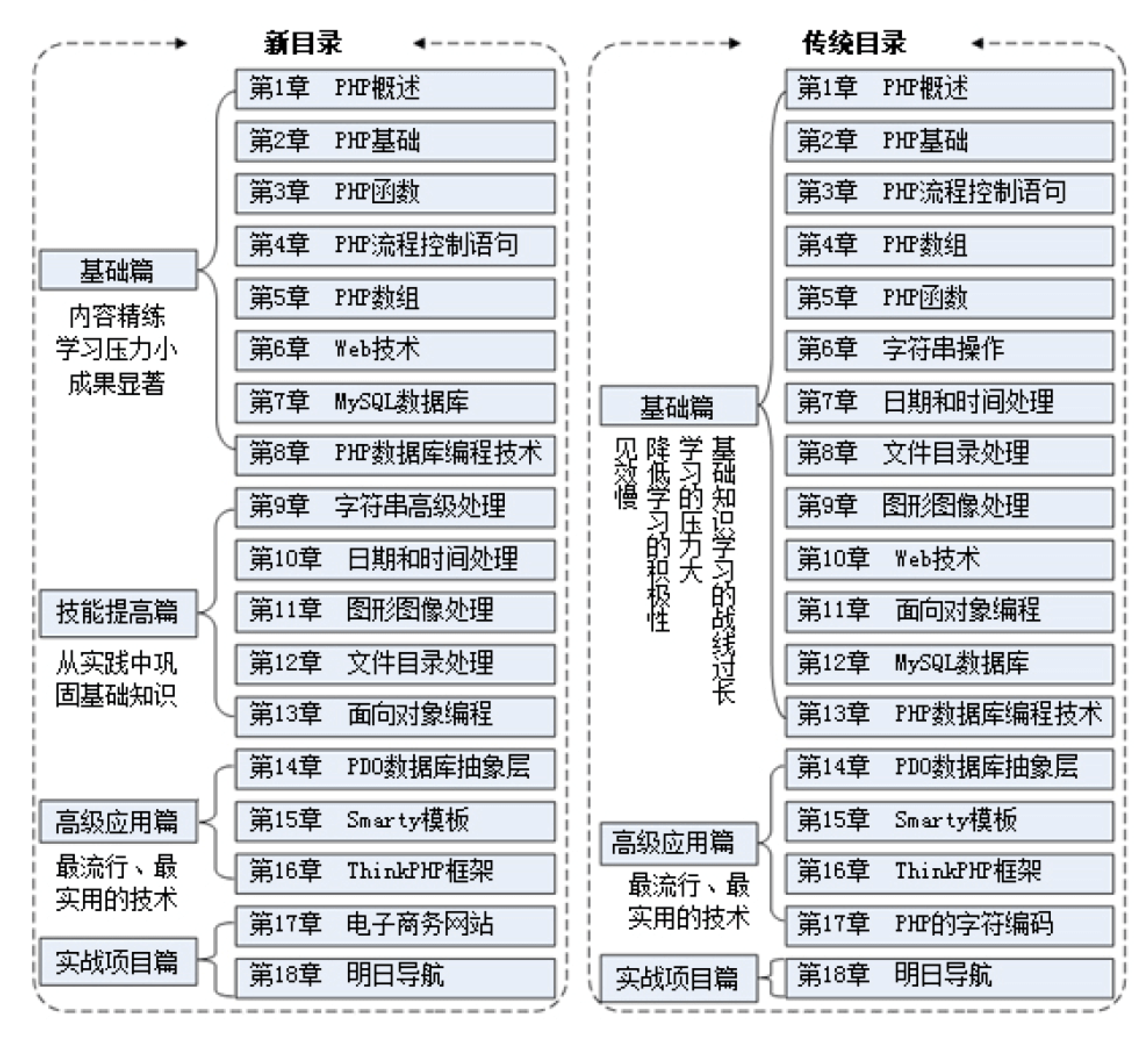


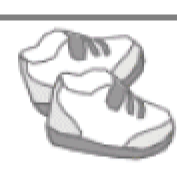
Preface

这是一次入门图书内容编排上的改革，它缩短了读者从初学者到达程序员的距离，增强了读者成为程序设计师的信心。希望通过此次改革，能够探索出一条更快、更好的适合初学者的学习之路。

本书内容

本书在目录安排上做了大幅度的调整，打破了传统目录的点、线、面一体化的结构布局，让读者能够更快进入项目开发的状态中去。我们对基础知识讲解的内容进行了精简，让读者提前了解 MySQL 数据库，掌握 PHP 操作 MySQL 数据库的方法，熟悉 PHP 项目开发的基本流程，更快地体会项目开发成功带来的快感，提高读者学习的积极性。本书章目录与传统章目录的对比效果如下图所示。







按照传统的目录结构学习，属于按部就班、逐一学习各项技术，其过程相当枯燥，很难保证读者有继续学习下去的积极性。

我们相信新的目录结构安排不会出现这样的问题，因为当读者学习了本书前 8 章内容之后，就已经体会到成功的快感了，并且能够在后续的技能提高、高级应用中逐步巩固基础知识，学习新技术，最终成为一个合格的 Web 项目程序设计师。

本书涵盖了 PHP 编程从入门到项目开发所必备的知识，并且分篇进行讲解，使读者的学习更有针对性。

第 1 篇　基础篇（第 1～8 章）：主要内容包括 PHP 概述、PHP 基础、PHP 函数、PHP 流程控制语句、PHP 数组、Web 技术、MySQL 数据库和 PHP 数据库编程技术。通过本篇的学习，读者能够熟练掌握 PHP 语言编程的基础知识，并能开发一些小型项目，体会 PHP 项目开发基本流程，感受成功的喜悦。

第 2 篇　技能提高篇（第 9～13 章）：主要内容包括字符串高级处理，日期和时间处理，图形图像处理，文件、目录处理和面向对象编程。通过本篇的学习，读者可以巩固所学基础知识，掌握更多的 PHP 开发技术。

第 3 篇　高级应用篇（第 14～17 章）：主要内容包括 PDO 数据库抽象层、Smarty 模板、ThinkPHP 框架和 PHP 的字符编码。通过本篇的学习，读者可以学习到 PHP 目前最流行、最实用的技术。

第 4 篇　实战项目篇（第 18 章）：通过模拟 hao123 网站，开发明日导航网站，以此来巩固所学的基础知识，掌握应用 ThinkPHP 框架开发 Web 项目的精髓。书中按照项目功能阐述→系统功能结构设计→数据库设计→创建项目→项目发布的过程进行介绍，带领读者逐步亲身体验开发项目的全过程。

本书特点

☑　技术新颖，讲解细致

全面、细致地展示 PHP 的基础知识，融入最新的 PDO 数据库抽象层、Smarty 模板和 ThinkPHP 框架技术等内容，让读者能够真正学习到 PHP 最流行、最实用的技术。

☑　实例丰富，贴近实际

在详细地讲解每个技术点的同时，以大量的示例和实例作为辅助，加深读者对知识的掌握，提高实践动手的能力。

☑　动手实践，一体学习

每章都以上机演练和实战模拟栏目的形式，为读者提供丰富的实战练习题目，让读者能够亲自动手实践，体验编程带来的成就感，并且将答案放置于配套资源中。另外，读者还可以参考《PHP 经典编程 265 例》一书，其中对本书上机演练和实战模拟栏目中所列的命题进行了完美的诠释。

☑　注释详尽，视频讲解

为了便于读者更好地学习和使用本书，书中所有的代码都提供了详尽的注释，而且配套资源中提供了覆盖全书的语音视频讲解，读者可以通过视频快速、直观、轻松地学习。



Note

本书配套资源

本书提供了内容丰富的配套资源，包括自学视频、源程序、素材，以及模块库、案例库、题库、素材库等多项辅助内容，读者朋友可以通过如下方式获取。

第 1 种方式：

（1）登录 www.tup.com.cn，在网页右上角的搜索文本框中输入本书书名（注意区分大小写和留出空格），或者输入本书关键字，或者输入本书 ISBN 号（注意去掉 ISBN 号间隔线“-”），单击“搜索”按钮。

（2）找到本书后单击超链接，在该书的网页下侧单击“网络资源”超链接，即可下载。

第 2 种方式：

访问本书的新浪微博 PHPbook，找到配套资源的链接地址进行下载。

读者对象

- ☑ 大中专院校师生
- ☑ 零基础学习人员
- ☑ 初中级程序开发人员
- ☑ 编程爱好者
- ☑ 面临就业的学生
- ☑ 相关培训机构的老师和学员
- ☑ 准备从事软件开发工作的求职者
- ☑ 立志编程的其他专业人士

读者服务&本书勘误

本书的“习题”提供了答案，可以从如下所述方式获得。

读者在使用本书过程中遇到的所有问题，均可通过以下方式联系我们。

1．新浪微博：PHPbook。

及时发布读者答疑、本书勘误、配套资料更新等内容。

2．腾讯 QQ：4006751066。

3．登录网站：www.mingribook.com，在论坛、勘误发布、读者纠错、技术支持、读者之家等栏目中的相关模块中提问、留言或查看。

本书作者

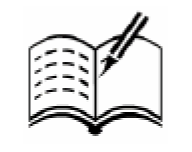
本书由明日科技组织编写，主要编写人员有潘凯华、李慧、刘欣、王小科、赵会东、聂喜婷、宋环雨、李继业、高春艳、赛奎春、杨丽、肖鑫、刘龄龄、张振坤、孙秀梅、王国辉、陈丹丹、陈英、朱晓和刘冠男等。在编写本书的过程中，我们以科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。

最后，感谢您选择本书，希望本书能成为您学习 PHP 编程路上的领航者。

编　者

目　录

Contents

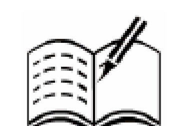


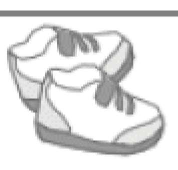
第 1 篇　基础篇



Note




Note



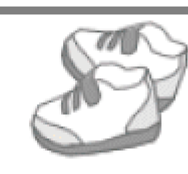

第 2 篇 技能提高篇



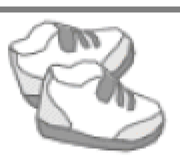

Note




第 3 篇 高级应用篇









第 4 篇 实战项目篇

▶▶第 1 篇

基础篇



PHP 概述

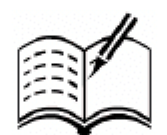
（ 自学视频、源程序：配套资源\mr\1\）

PHP 是一种服务器端、跨平台、面向对象、HTML 嵌入式的脚本语言。本章将向读者简单介绍 PHP 语言、PHP 的语言优势、下载 PHP 及相关软件、搭建 PHP 的开发环境，了解常用的配置信息，熟悉 PHP 开发环境的配置结构等知识。其主要目的是让读者先在宏观上对 PHP 语言有一个整体的了解，使读者对 PHP 有一个总体的认识，找到学习 PHP 的切入点。

学习摘要：

- 如何学好 PHP
- AppServ——Windows 版 PHP 集成化安装包
- XAMPP——Linux 版 PHP 集成化安装包
- PHP 开发环境的关键配置信息
- 解决 PHP 的常见配置问题
- 通过实例了解 PHP 配置环境的相关信息





1.1 如何学好 PHP

1.1.1 什么是 PHP

PHP 是 Hypertext Preprocessor（超文本预处理器）的缩写，是一种服务器端、跨平台、面向对象、HTML 嵌入式的脚本语言。其独特的语法混合了 C 语言、Java 语言和 Perl 语言的特点，是一种被广泛应用的开源式的多用途脚本语言，尤其适合 Web 开发。图 1.1 所展示的就是代表 PHP 语言的图标。



图 1.1 PHP 图标

1.1.2 PHP 版本

PHP 于 1994 年由 Rasmus Lerdorf 创建。最初只是一个简单的用 Perl 语言编写的程序，用来统计网站的访问者。后来又用 C 语言重新编写，增加了多种功能，包括可以访问数据库等。1995 年以 Personal Home Page Tools (PHP Tools)为名开始对外发表第一个版本，Lerdorf 编写了一些介绍此程序的文档，并且发布了 PHP1.0。在早期的版本中，提供了访客留言本、访客计数器等简单功能。后来越来越多的网站使用 PHP，并且强烈要求增加一些特性，如循环语句和数组变量等，在新的成员加入到开发行列之后，在 1995 年中又发布了 PHP2.0，定名为 PHP/FI (Form Interpreter)。PHP/FI 加入了对 mSQL 的支持，从此建立了 PHP 在动态网页开发上的地位。到 1996 年底，大约有 15000 个网站在使用 PHP/FI；1997 年中，使用 PHP/FI 的网站总数超过 5 万个。而在 1997 年中，开始了第三版的开发计划，开发小组加入了 Zeev Suraski 及 Andi Gutmans，而第三版就定名为 PHP3。

【PHP4】

2000 年，PHP4.0 问世，其中增加了许多新的特性。PHP4.0 整个脚本程序的核心大幅变动，加快了程序的执行速度，满足了更快的要求。其在最佳化之后的效率，已较传统 CGI 或 ASP 等程序有更好的表现，而且还有更强的新功能、更丰富的函数库。事实证明，PHP 在 Web CGI 的领域上掀起了巅覆性的革命。对于一位专业的 Web Master 而言，它将也是必修课程之一。

【PHP5】

在 PHP 的发展过程中又推出了 PHP5，其功能更加完善，很多缺陷和 BUG 都被逐一修复。在 PHP 5 中，理想的选择是 PHP5.2.X 系列，其兼容性好，每次版本的升级带来的都是安全性和稳定性的改善。但如果产品是自己开发使用，则 PHP5.3.X 在某些方面更具优势，在稳定性上更胜一筹，增加了很多 PHP5.2.X 所不具有的功能，如内置 php-fpm、更完善的垃圾回收算法、命名空间的引入、sqlite3 的支持等，是部署项目值得考虑的版本（本书中使用是 PHP5 版本）。

【PHP6】

时至今日，PHP 的版本已经更新到 PHP6。PHP6 是一个理想化的产品，虽然目前还没有走上生产线，但其更新的特性和功能极大地吸引着广大的 PHP 编程爱好者。

☑ 完全抛弃全局变量。

☑ 删除 Magic Quotes。

☑ 增加一个输入过滤扩展代替 Magic Quotes，提供一个机制让开发者很容易自己关闭或




开启这个功能，而不是像现在这样先判断服务器的 GPC 是否打开。

☑ 默认加入 opcode cache，对代码执行进行速度上的优化。目前大多用的是 PECL 或 APC，但有一个官方的解决方案显然是比较好的。
☑ 删除安全模式 safe_mode，改进 open_basedir。
☑ 删除在 PHP3/4 中已经被标记为过时 deprecated 的内容。
☑ 标识符（程序中使用的变量名、函数名、标号等）大小写敏感。
☑ 删除各种函数的别名。




1.1.3 PHP 的应用领域

在互联网高速发展的今天，PHP 的应用领域可谓是非常广泛，主要包括：

☑ 中小型网站的开发。
☑ 大型网站的业务逻辑结果展示。
☑ Web 办公管理系统。
☑ 硬件管控软件的 GUI。
☑ 电子商务应用。
☑ Web 应用系统开发。
☑ 多媒体系统开发。
☑ 企业级应用开发。

PHP 正吸引着越来越多的 Web 开发人员，其无处不在，它可应用于任何地方、任何领域，并且已拥有几百万个用户，其发展速度要快于在它之前的任何一种计算机语言。PHP 能够给企业和最终用户带来无穷无尽的好处。据最新数据统计，全世界有超过 2200 万家的网站和 1.5 万家公司在使用 PHP 语言，包括百度、雅虎、Google、YouTube、Digg 等著名的网站，也包括汉沙航空电子订票系统、德意志银行的网上银行、华尔街在线的金融信息发布系统等，甚至包括军队系统这类苛刻的环境。除此之外，PHP 也是企业用来构建服务导向型、创造和混合 Web 融于新一代的综合性商业应用的语言，其也逐渐发展成为开源商业应用发展的方向。PHP 的成功应用案例如图 1.2 和图 1.3 所示。



图 1.2　百度网页



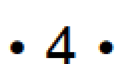
图 1.3　谷歌网页

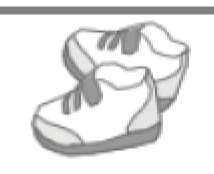
多学两招：

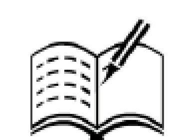
虽然已经推出 PHP6，但是目前应用的主流依然是 PHP5。

1.1.4 PHP5 的新特性

目前主流仍然是 PHP5，所以下面着重讲述 PHP5 中新的对象模型的特性。







- ☑ 构造函数和析构函数。
- ☑ 对象的引用。
- ☑ 对象的克隆（clone）。
- ☑ 对象中的私有、公共及受保护模式（public/private 和 protected 关键字）。
- ☑ 接口（Interface）。
- ☑ 抽象类。
- ☑ __call。
- ☑ __set 和__get。
- ☑ 静态成员。

1.1.5　下载 PHP 及相关软件

搭建 PHP 环境涉及到系统平台、Web 服务软件和数据库软件及 PHP 本身，用户可根据自身现有的计算机软、硬件环境，自由选择相应的软件。

通常选择 Windows NT 为实验平台，这样可以下载 PHP5.0 以上的 Windows 版本。数据库可以下载 MySQL 的 Windows 版本（www.MySQL.org）或者使用微软公司的 MsSQL。Web 服务软件可以直接下载 Apache 的 Windows 版本（www.apache.com）。

如果想搭建 Linux 下运行的实战环境，那么所有这些软件必须下载其对应于 Linux 的版本，有的可能需要在 Linux 下编译生成。

下面笔者以 Windows 版本为例，简单说明一下 PHP 优秀的集成开发环境及相关信息。

（1）XAMPP 是一个易于安装且包含 MySQL、PHP 和 Perl 的 Apache 发行版，只需根据提示操作，即可安装成功。而不必对 PHP、Apache、MySQL 配置文件进行修改及相关烦琐的操作（例如，将 PHP 的配置文件 php.ini 保存到操作系统 C 盘下的 WINDOWS 文件夹下，手动开启 MySQL 组件、Oracle 组件和 GD2 支持等），大大节省了初学者在配置运行环境时的时间，真正意义上做到了一键安装、开发运行的理念。XAMPP 的官方下载地址如图 1.4 所示。

（2）AppServ 将 Apache、PHP、MySQL 和 phpMyAdmin 等服务器软件和工具安装配置完成后打包处理，同 XAMPP 一样，安装相当简单，其官方下载地址如图 1.5 所示。

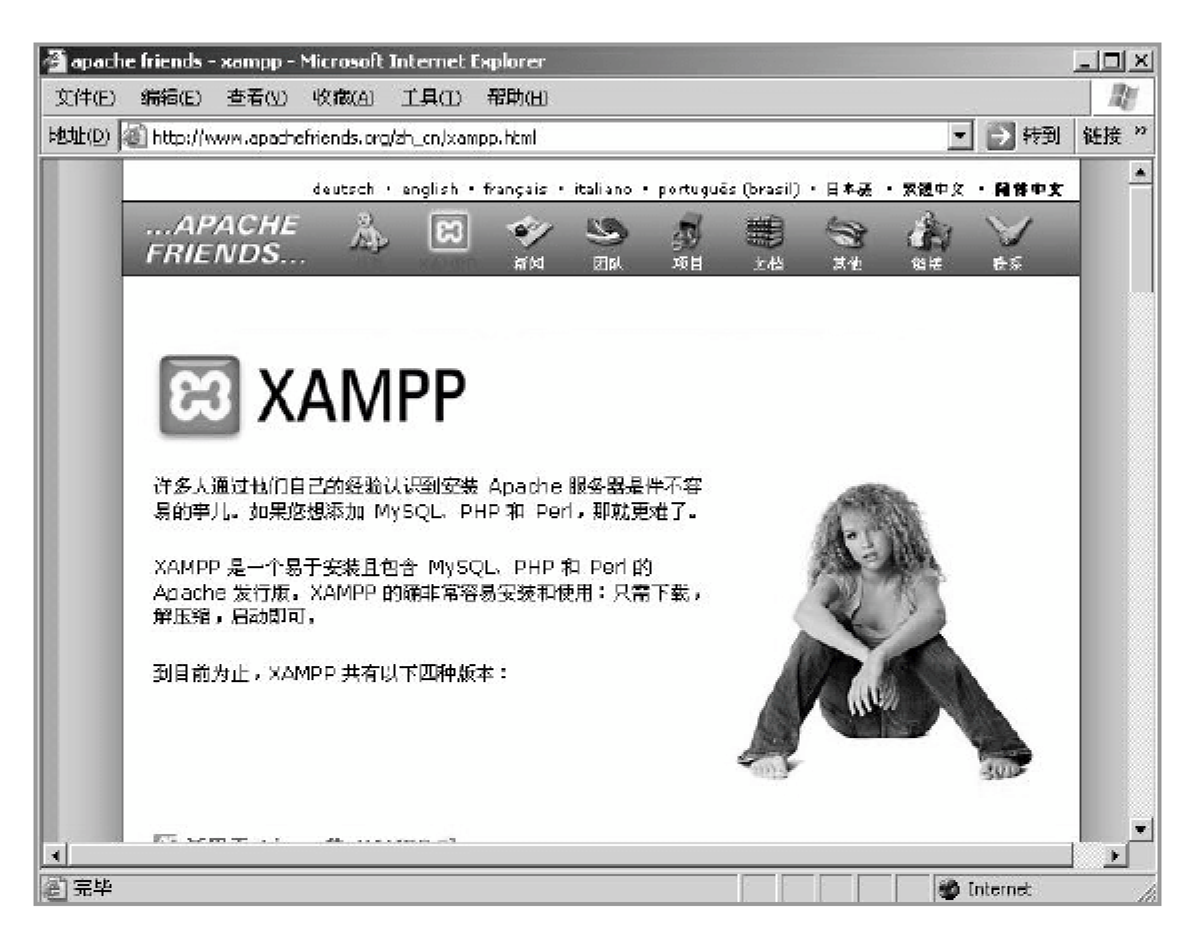
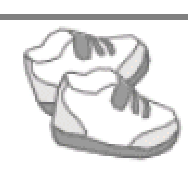


图 1.4　XAMPP 开发环境的下载截图



图 1.5　AppServ 开发环境的下载截图





Note

1.1.6　代码编辑工具

选择 PHP 的代码编辑工具，笔者认为应该考虑 4 个方面的因素：

第一，语法的高亮显示。应用语法的高亮显示，可以对代码中的不同元素采用不同的颜色进行显示，例如，关键字用蓝色，对象方法用红色标识等。

第二，格式排版功能。该功能可以使程序代码的组织结构清晰易懂，并且易于程序员进行程序调试，排除程序的错误异常。

第三，代码提示功能。该功能可以在程序员编写某个函数时，提供这个函数的语法信息，甚至可以在程序员输入某个字符时，给出这个字符相关的函数信息，从而帮助程序员编写正确的函数，使用正确的语法。

第四，界面设计功能。利用该功能不但可以编写 PHP 代码，还可以进行界面的设计。

以上是在选择代码编辑工具时，用户应考虑的问题，需要注意的是，这 4 个因素不可能都完全满足，用户应根据自己的实际情况进行选择。

下面介绍几款常用的代码编辑工具，供广大读者参考。

☑　Macromedia Dreamweaver

Macromedia Dreamweaver 是一款专业的网站开发编辑器，它将可视布局工具、应用程序开发功能和代码编辑支持组合在一起，其功能强大，使得各个层次的开发人员和设计人员都能够快速创建出吸引人的、标准的网站和应用程序。Macromedia Dreamweaver 采用了多种先进的技术，能够快速、高效地创建极具表现力和动感效果的网页，使网页创作过程简单无比，同时，它还提供了代码自动完成功能，不但可以提高编写速度，而且还减少了错误代码出现的几率。Macromedia Dreamweaver 既适用于初学者制作简单的网页，又适用于网站设计师、网站程序员开发各类大型应用程序，极大地方便了程序员对网站的开发与维护。

Macromedia Dreamweaver 从 MX 版本开始支持 PHP+MySQL 的可视化开发，对于初学者确实是比较好的选择，因为如果是一般性开发，几乎是可以不写一行代码也可以写出一个程序，而且都是所见即所得的。其所包含的特征包括：语法加亮、函数补全、形参提示、全局查找替换、处理 Flash 和图像编辑等。同时，它还为 PHP、ASP 等脚本语言提供辅助支持。

下载地址：http://www.adobe.com/downloads/。

☑　ZendStudio

ZendStudio 是目前公认的最强大的 PHP 开发工具，是一款收费软件。其具备功能强大的专业编辑工具和调试工具，包括：编辑、调试，配置 PHP 程序所需要的客户及服务器组件，支持 PHP 语法加亮显示，尤其是功能齐全的调试功能，能够帮助程序员解决在开发中遇到的很多问题。

下载地址：http://www.zend.com/store/products/zend-studio.php。

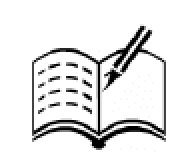
说明：

ZendStudio 虽然是收费软件，但如果是个人使用，则可以下载其试用版。

☑　PHPEdit

PHPEdit 是一款 Windows 下优秀的 PHP 脚本 IDE（集成开发环境）。该软件为快速、便捷的开发 PHP 脚本提供了多种工具。其功能包括：语法关键词高亮；代码提示、浏览；集成 PHP





调试工具；帮助生成器；自定义快捷方式；150 多个脚本命令；键盘模板；报告生成器；快速标记；插件等。

官方网址：http://phpedit.svoi.net。

1.1.7 下载 PHP 用户手册

关于获取 PHP 帮助信息的途径，除了可以到书店购买 PHP 的相关教材外，还可以直接到 PHP 的官方网站中下载 PHP 中文手册，PHP 机构的官方网址为 http://www.php.net，初学者可以在该手册中查找到相关函数的详细说明。由于 PHP 代码公开，而且完全免费，所以用户可以直接在 PHP 的官方网站中下载 PHP 中文手册。

PHP 中文手册是学习 PHP 的良师，许多 PHP 的高手仍然随时备查。该手册通常为 PHP5.0，有中、英文版本，中文版中还有 HTML、ZIP 和 CHM 等几种格式，目前很多网站都可以进行下载。笔者建议下载 CHM 格式，查阅较为方便。

读者在学习的过程中可以通过下载 PHP 中文手册和 MySQL 中文手册来获得帮助。为了便于学习，下面推荐两个网址供参考。

（1）PHP 中文手册下载地址：http://www.php.net。PHP 中文手册的运行效果如图 1.6 所示。

（2）MySQL 中文手册下载地址：http://www.mysql.com。MySQL 中文手册的运行效果如图 1.7 所示。

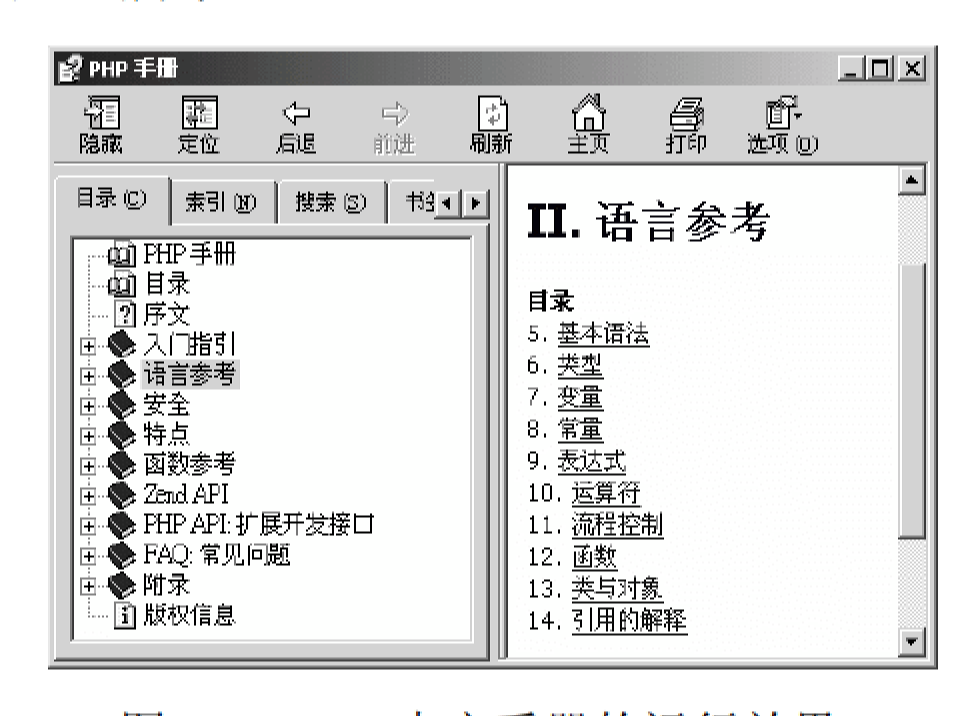


图 1.6 PHP 中文手册的运行效果

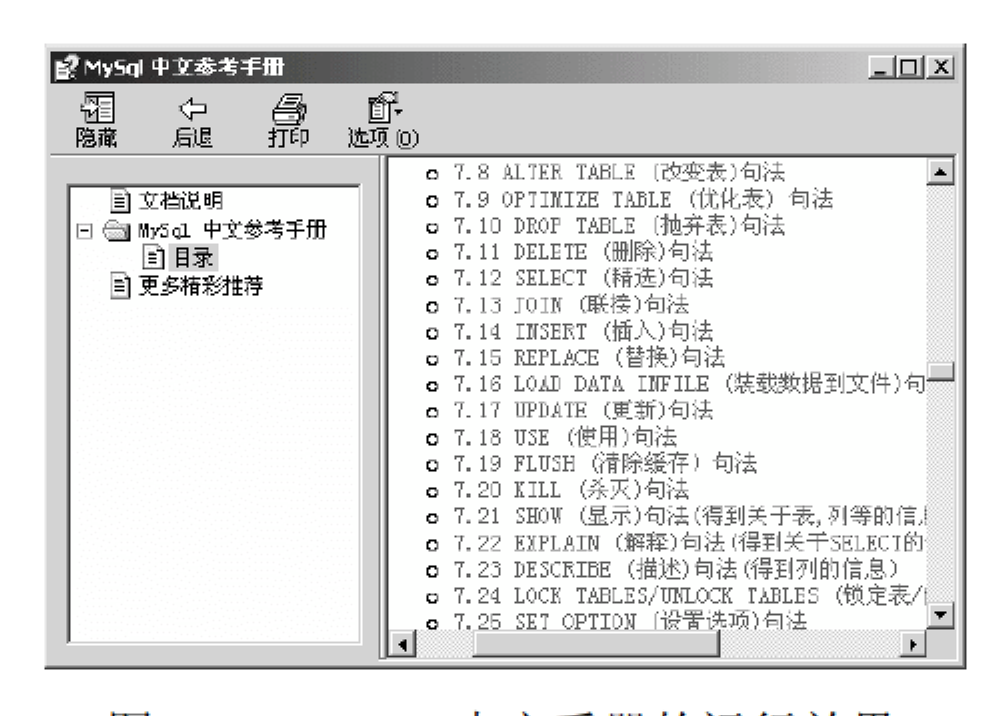


图 1.7 MySQL 中文手册的运行效果

1.2 环境的搭建

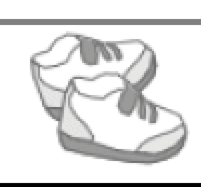
视频讲解：配套资源\mr\1\video\AppServ 搭建 PHP 环境.exe

1.2.1 AppServ——Windows 版 PHP 集成化安装包

AppServ 将 Apache、PHP、MySQL 和 phpMyAdmin 等服务器软件和工具安装配置完成后打包处理，开发人员只要到网站上下载该软件并安装，即可完成 PHP 开发环境的快速搭建，非常适合初学者使用。

脚下留神：

在使用 AppServ 搭建 PHP 开发环境时，必须确保系统中没有安装 Apache、PHP 和 MySQL，否则，要先将这些软件卸载，然后应用 AppServ。





Note

指点迷津：

PHP 开发环境集成安装包：目前网络上流行的集成安装包有十几种，其中比较常用是 EasyPHP、AppServ 和 XAMPP，它们都可以搭建 PHP 的开发环境。但是，根据实际的应用情况，还是 AppServ 相对比较好用一些，因为诸如 EasyPHP 或 XAMPP，在对一些类库的支持上会产生一些错误，导致类库不能正常使用。例如，在 EasyPHP 或 XAMPP 中应用 jpgraph 类库开发图像时，就会出错。

下面讲解 AppServ 集成化安装包搭建 PHP 开发环境的具体操作步骤：

（1）双击 AppServ-win32-2.5.10.exe 文件，打开如图 1.8 所示的 AppServ 启动页面。

（2）单击图 1.8 中的 Next 按钮，打开如图 1.9 所示的 AppServ 安装协议页面。

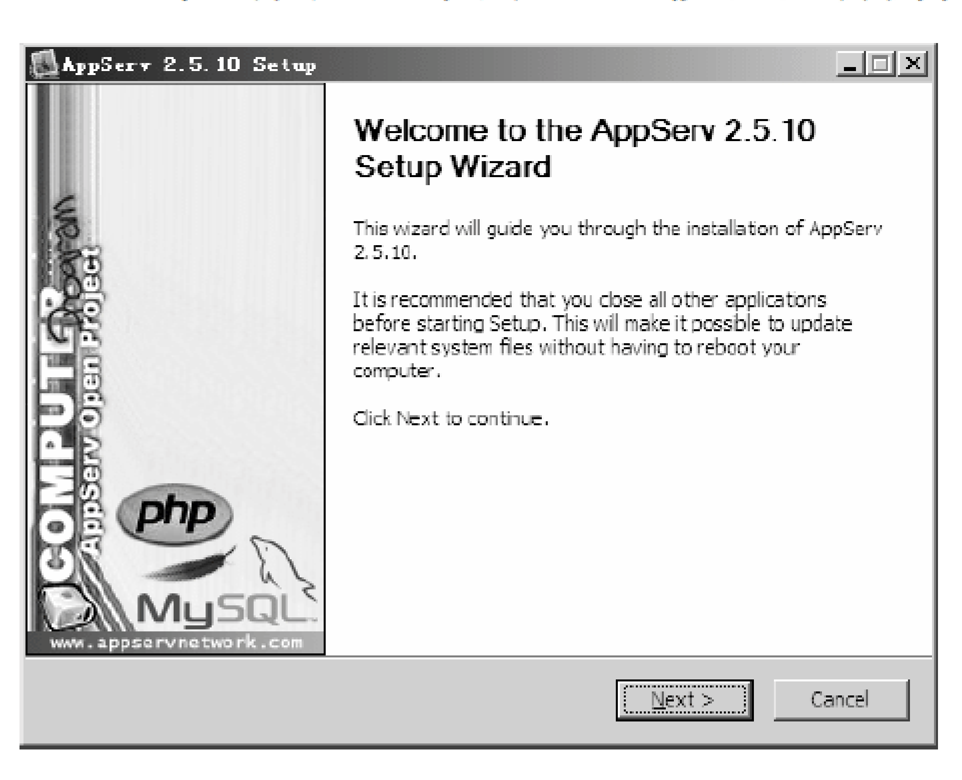


图 1.8　AppServ 启动页面

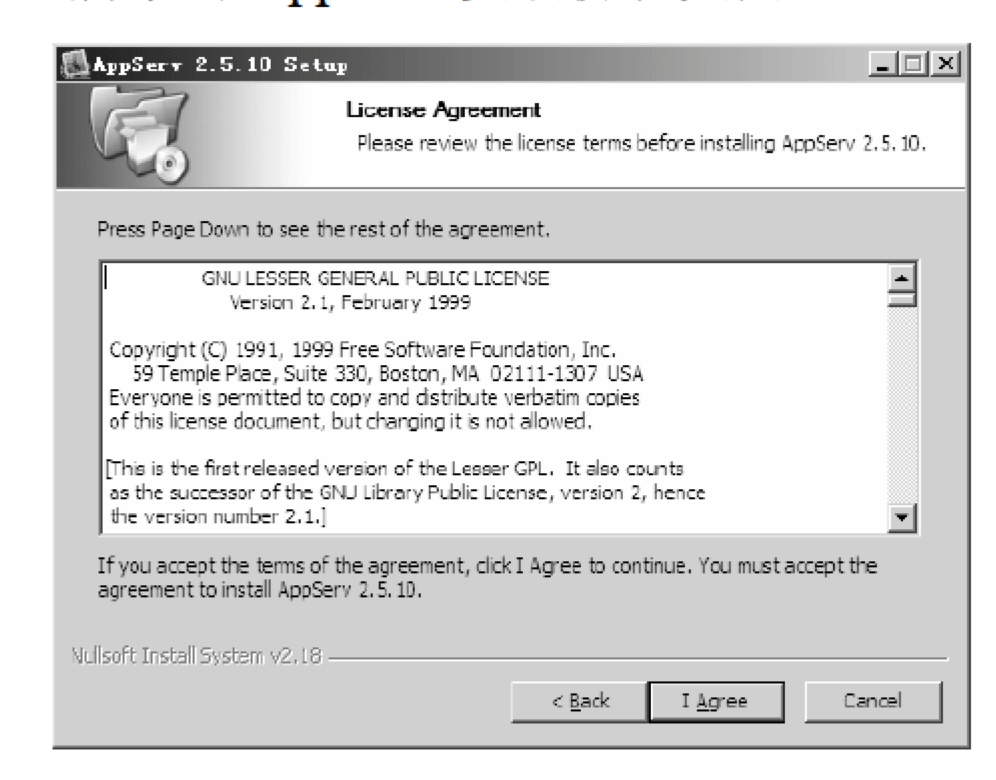


图 1.9　AppServ 安装协议

（3）单击图 1.9 中的 I Agree 按钮，打开如图 1.10 所示的页面。设置 AppServ 的安装路径（默认安装路径一般为：E:\AppServ），当 AppServ 安装完成后，Apache、MySQL、PHP 都将以子目录的形式存储到该目录下。

（4）单击图 1.10 中的 Next 按钮，打开如图 1.11 所示的页面。选择要安装的程序和组件（默认为全选）。

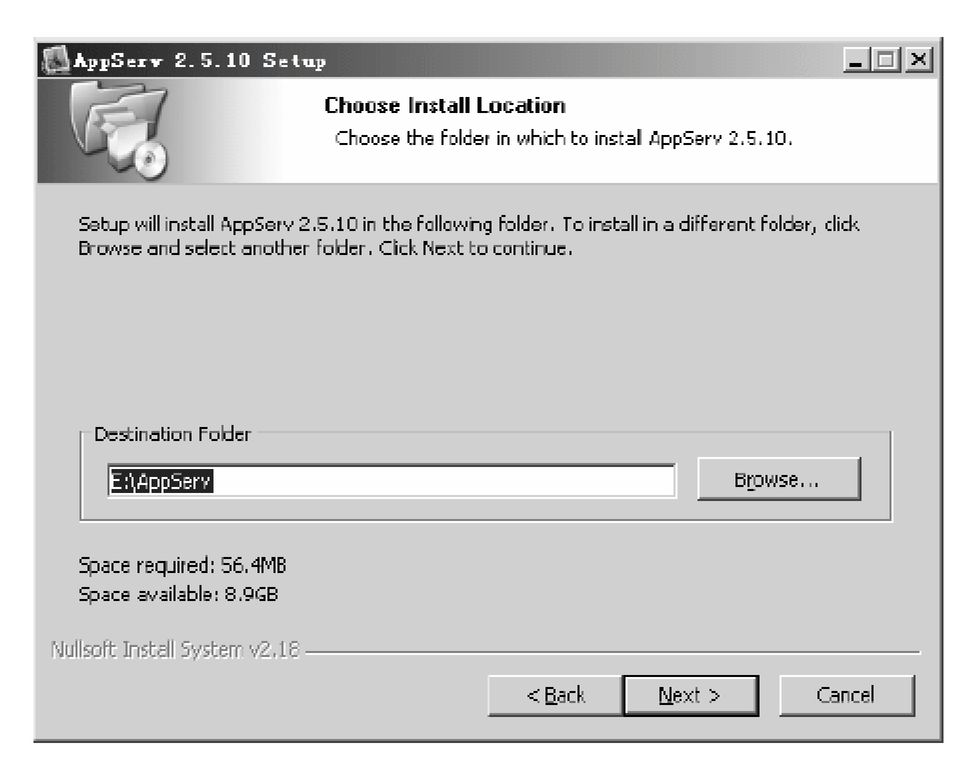


图 1.10　AppServ 安装路径选择

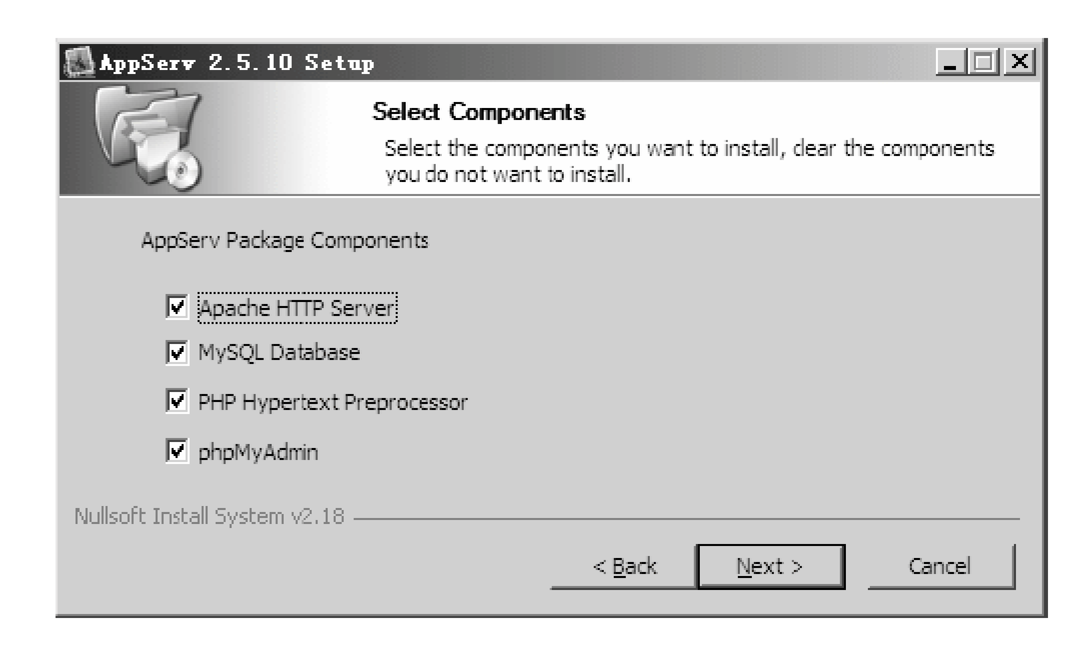


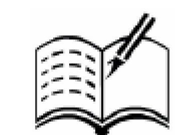
图 1.11　AppServ 安装选项

脚下留神：

在图 1.11 的操作步骤中，如果本机中已经安装了 MySQL 数据库，那么此时可以不选中 MySQL Database 复选框，仍使用本机已经存在的 MySQL 数据库。



（5）在图 1.11 中单击 Next 按钮，打开如图 1.12 所示的页面。填写计算机名称，添加邮箱地址，设置 Apache 的端口号，默认为 80 端口。



多学两招：

Apache 服务器端口号的设置直接关系到 Apache 服务器是否能够正常启动。如果本机中的 80 端口被 IIS 或迅雷占用，那么此时就需要修改 Apache 的端口号，或者将 IIS、迅雷的端口号修改，才能完成 Apache 服务器的配置。如果出现端口冲突，那么将导致安装失败，Apache 服务不能启动。

（6）单击图 1.12 中的 Next 按钮，打开如图 1.13 所示的页面。设置 MySQL 数据库 root 用户的登录密码及字符集。

多学两招：

MySQL 数据库字符集的设置，可以选择 UTF-8、GBK 或 GB2312。这里将字符集设置为“UTF-8 Unicode”，表示 MySQL 数据库的字符集将采用 UTF-8 编码。

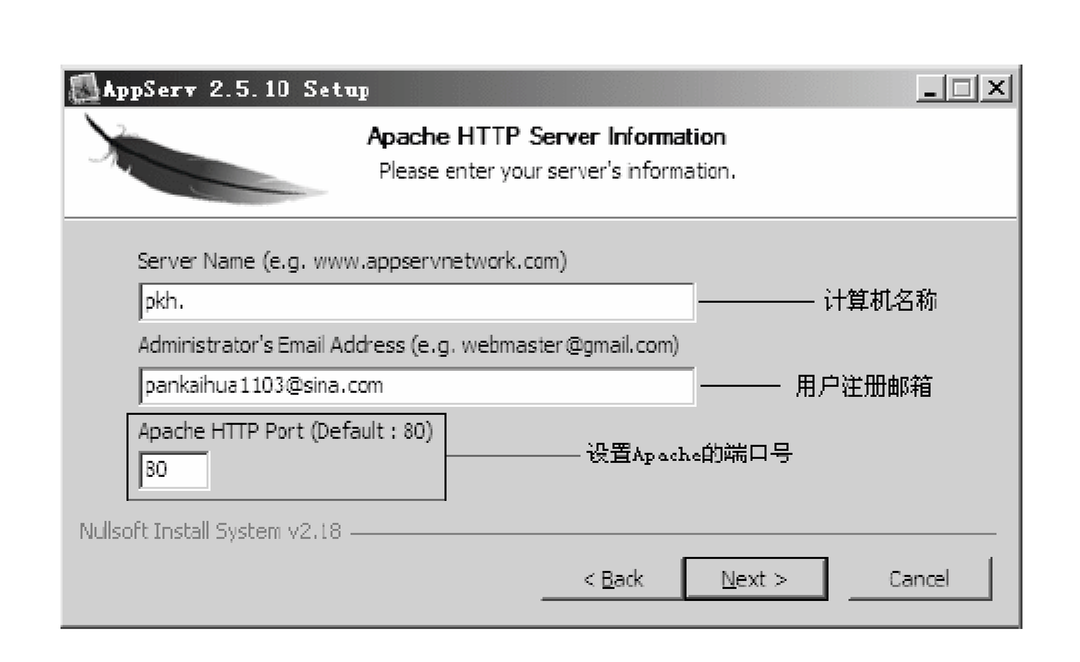


图 1.12 Apache 端口号设置

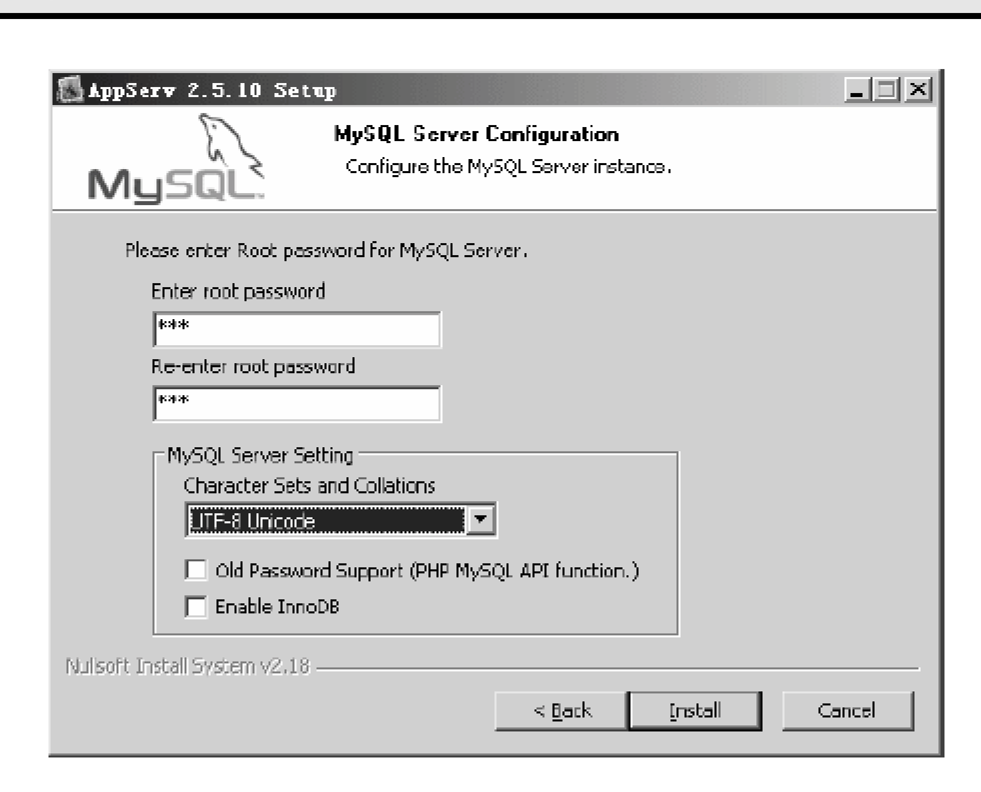


图 1.13 MySQL 设置

多学两招：

对于在图 1.13 中设置的 MySQL 数据库 root 用户的密码必须牢记，因为程序在连接数据库时必须使用这个密码。建议读者将这个密码设置为 111，因为这是在开发本书中的程序时所使用的数据库密码。如此，当读者在运行本书中的数据库程序时就不需要修改密码；否则，当运行本书的程序时，要修改连接数据库的密码。如果忘记安装时设置的密码，最直接有效的解决方式是重新安装 AppServ。

（7）单击图 1.13 中的 Install 按钮开始安装，如图 1.14 所示。

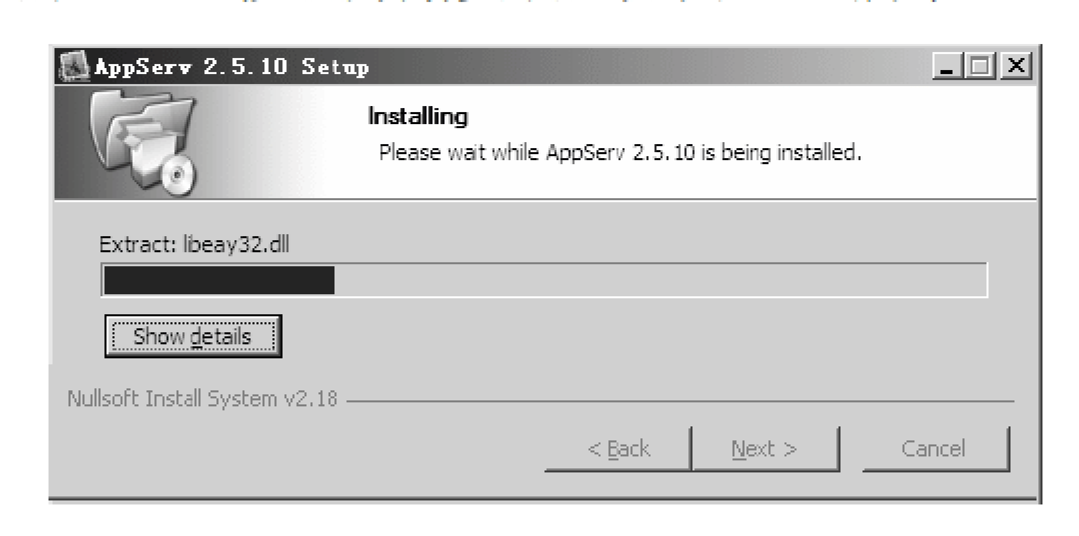
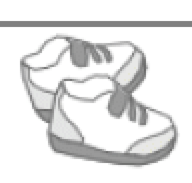
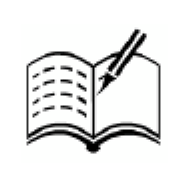


图 1.14 AppServ 安装页面





（8）安装完成后可以在开始菜单的 AppServ 相关操作列表中启动 Apache 及 MySQL 服务，如图 1.15 所示。

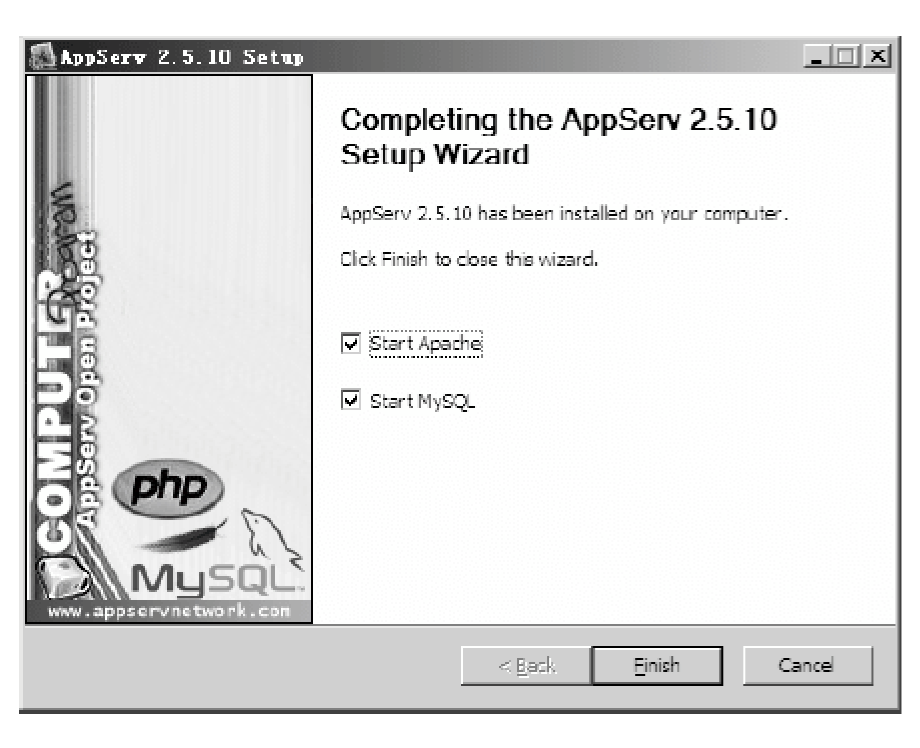


图 1.15　AppServ 安装完成页面

1.2.2　XAMPP——Linux 版 PHP 集成化安装包

本节中主要介绍在 Linux 操作系统下通过 XAMPP 配置 PHP 的开发环境及其基本应用。XAMPP（Apache+MySQL+PHP+PERL）是一个功能强大的建站集成软件包，它可以在 Windows、Linux、Solaris 3 种操作系统下安装使用，支持多语言，包括英文、简体中文、繁体中文、韩文、俄文、日文等。

在 Linux 操作系统下，使用集成软件 XAMPP 的 Linux 版来配置 PHP 开发环境。Linux 下安装 XAMPP 的步骤如下：

（1）在 Linux 操作系统下，选择“主菜单”/“系统工具”/“终端”命令。

（2）在命令模式下，首先进入到系统的根目录下。

（3）通过 mkdir 命令在根目录下创建一个 opt 目录。

（4）通过 tar xvfz 命令将 XAMPP 解压缩到 opt 目录下。

脚下留神：

在 Linux 操作系统下执行 XAMPP 解压缩的命令时，必须进入到系统的根目录下，并且要在根目录下创建一个子目录，然后才可以执行解压缩的命令；否则，将提示无法找到指定的目录或文件。

（5）按 Enter 键，执行 XAMPP 的解压缩，直到安装成功。其具体使用的命令如图 1.16 所示。

（6）安装成功后，查看/opt/lampp 目录，如图 1.17 所示。

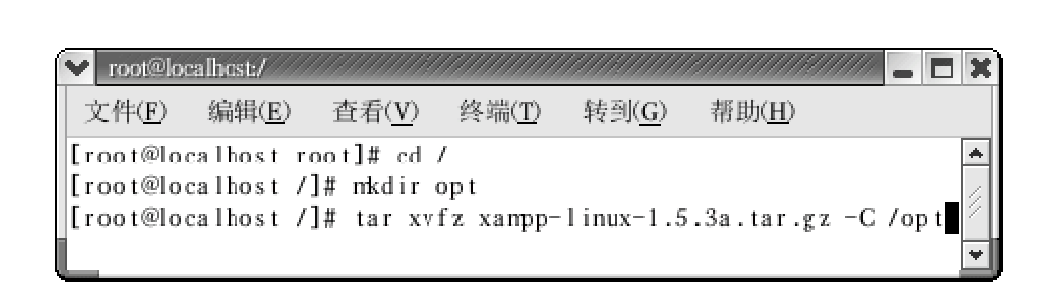


图 1.16　XAMPP 的安装

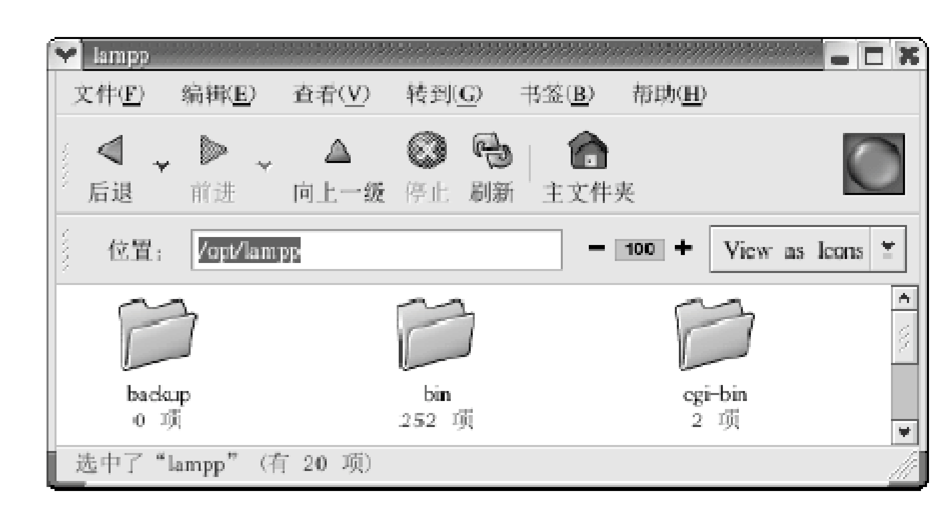


图 1.17　查看安装的内容



（7）在 Mozilla 浏览器中输入 http://127.0.0.1/xampp/index.php，运行结果如图 1.18 所示。

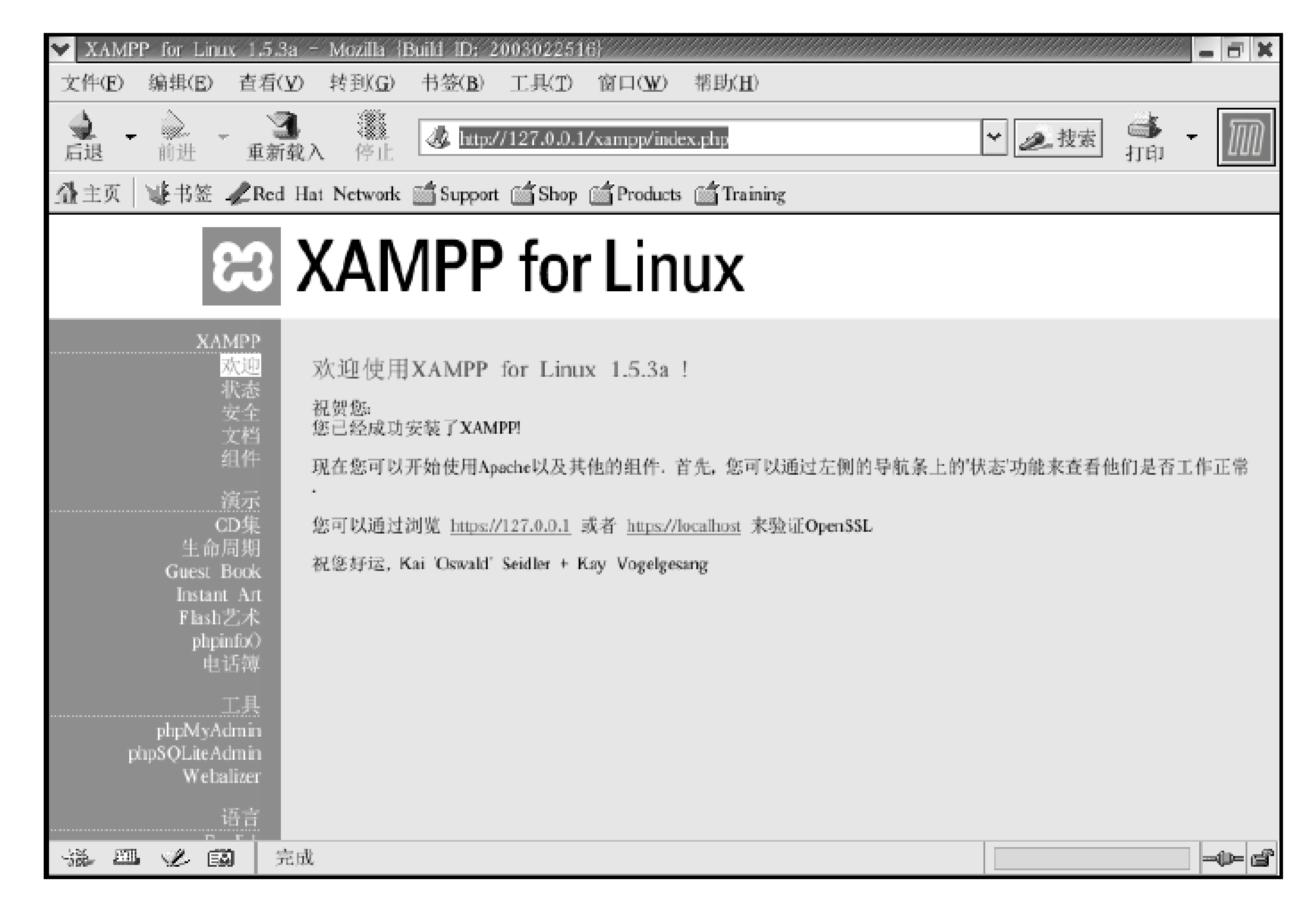



图 1.18　Linux 版的 XAMPP

脚下留神：

在 Linux 操作系统下安装 XAMPP，必须在 Linux “主菜单” / “系统工具” 下的 “终端” 命令中通过指令来完成，切记不要使用任何微软操作系统下的工具来进行操作。

多学两招：

Linux 下卸载 XAMPP 的命令：rm –rf /opt/lampp。

上机演练

上机演练 1　测试 AppServ 是否安装成功

测试 AppServ 是否安装成功的方法为：打开 IE 浏览器，在地址栏中输入 http://localhost/ 或 http://127.0.0.1/，如果打开如图 1.19 所示的页面，则说明 AppServ 安装成功。

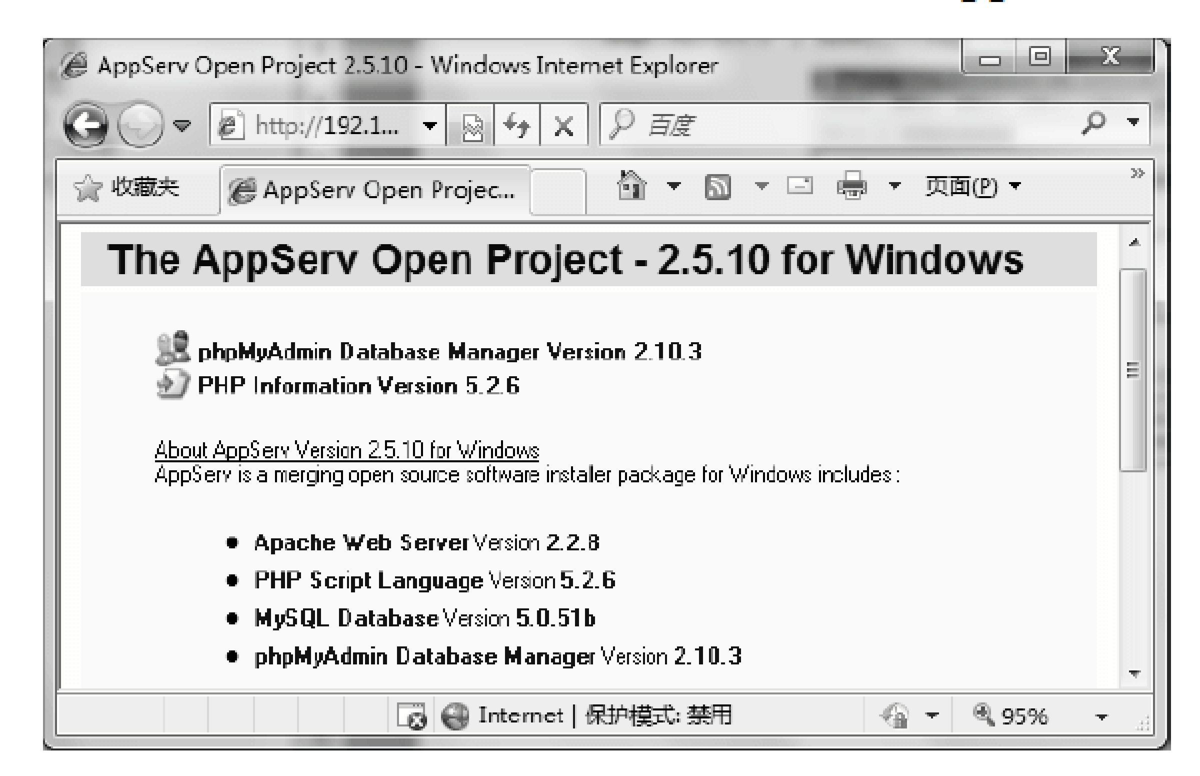
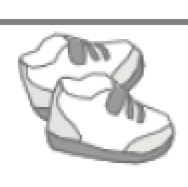


图 1.19　AppServ 测试页

上机演练 2　第一次登录 phpMyAdmin 图形化管理工具

AppServ 安装完成并测试成功后，打开 IE 浏览器，在地址栏中输入 http://localhost/或





Note

http://127.0.0.1/，打开如图 1.19 所示的页面，单击页面上方的 phpMyAdmin Database Manager Version2.10.3 超链接，打开如图 1.20 所示的页面，它是图形化管理工具 phpMyAdmin 的登录窗口，输入在安装 AppServ 时设置的 MySQL 服务器的用户名和密码（默认用户名为 root，本机设置的密码为 111），即可登录到 phpMyAdmin 图形化管理工具界面，如图 1.21 所示。

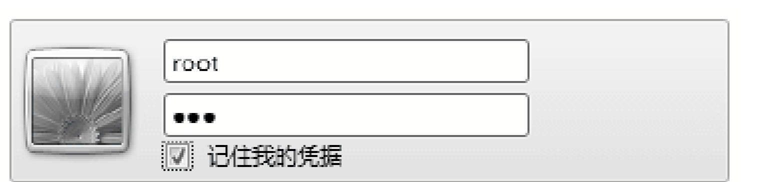


图 1.20　phpMyAdmin 的登录窗口



图 1.21　phpMyAdmin 图形化管理工具界面

1.3　PHP 开发环境的关键配置信息

视频讲解：配套资源\mr\1\video\PHP 开发环境的关键配置信息.exe

前面介绍了 PHP 开发环境的配置方法，除了安装步骤本身之外，PHP 与服务器的配置也十分重要。下面将主要介绍 PHP 及 Apache 服务器的配置。

1.3.1　Apache 服务器的基本配置

Apache 服务器的设置文件在 Linux 操作系统中位于/usr/local/apache/conf/（在 Windows 操作系统中位于/etc/httpd/conf）目录下，基本上使用以下 3 个配置文件来配置 Apache 服务器的行为。

☑　access.conf：用于配置服务器的访问权限，控制不同用户和计算机的访问限制。

☑　httpd.conf：用于设置服务器启动的基本环境。

☑　srm.conf：主要用于进行文件资源上的设定。

指点迷津：

http.conf 是 Apache 服务器的配置文件，其常用的配置包括：Apache 服务器的端口号、服务器的访问路径和伪静态的设置。

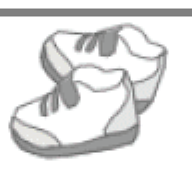
ServerName localhost:80

DocumentRoot "/xampp/htdocs"

LoadModule rewrite_module modules/mod_rewrite.so

1.3.2　php.ini 文件的基本配置

php.ini 文件是 PHP 在启动时自动读取的配置文件。它是一个 ASCLL 文本文件，分为多个部分，每一部分包括相关的参数。每一部分的名称位于最前面的方括号内，接着是名称对数字，



每一名称都独占一行。使用规则 PHP 代码，对参数名称非常敏感，不能包含有空格，但是参数可以是数字、字符串或布尔逻辑数。分号位于每一行的开始，作为指定标记，这就使选择使用或者不使用 PHP 的这些特性变得很方便，而无须通过删除该行来实现。对某特性进行注释（即添加分号），则该行将不会被编译执行。每次修改完 php.ini 文件，必须重新启动 Apache 服务器，以使新的设置生效。

指点迷津：

php.ini 是 PHP 的配置文件，用于加载各种函数库、设置错误级别和设置服务器的时间等。

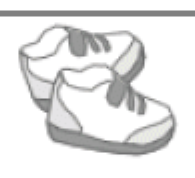
在 Linux 操作系统中，php.ini 存储于/opt/lampp/etc/php.ini 文件夹下，而在 Windows 操作系统中，php.ini 存储于系统盘的 windows 文件下。php.ini 文件的基本配置如表 1.1 所示。

表 1.1　php.ini 文件的基本配置

参　　数	说　　明	默　认　值
error_reporting	设置错误处理的级别。推荐值为 E_ALL & ~E_NOTICE &~E_STRICT，显示所有错误信息，除了提醒和编码标准化警告	E_ALL & ~E_NOTICE & ~E_STRICT
register_globals	通常情况下可以将此变量设置为 Off，这样可以对通过表单进行的脚本攻击提供更为安全的防范措施	register_globals = On
include_path	设置 PHP 的搜索路径，这一参数可以接收系列的目录。当 PHP 遇到没有路径的文件提示时，它将会自动检测这些目录，需要注意的是，当某些选项允许多个值时，应使用系统列表分隔符，在 Windows 下使用分号“;”，在 Linux 下使用冒号“:”	; UNIX: "/path1:/path2" ;include_path = ".:/php/includes" ; Windows: "\path1;\path2" ;include_path = ".;c:\php\includes"
extension_dir	指定 PHP 的动态连接扩展库的目录	\ext 目录下
extension	指定 PHP 启动时所加载的动态连接扩展库。PHP 的常用扩展库及其说明如表 1.2 所示	PHP 的常用扩展库在初次安装配置后均被注释，需读者手动更改
file_uploads	设置是否允许通过 HTTP 上传文件	file_uploads=On
upload_tmp_dir	设置通过 HTTP 上传文件时的临时目录，如果为空，则使用系统的临时目录	upload_tmp_dir =空
upload_max_filesize	设置允许上传文件的大小，如 50M，必须填写单位	upload_max_filesize=2M
post_max_size	控制在采用 POST 方法进行一次表单提交中 PHP 所能够接收的最大容量。要上传更大的文件，则该值必须大于 upload_max_filesize 的值。如 upload_max_filesize=10M，那么 upload_max_filesize 的值必须要大于 10M	post_max_size = 8M
max_input_time	以秒为单位对通过 POST、GET 以及 PUT 方式接收数据时间进行限制	max_input_time = 60

表 1.2　PHP 常用扩展库及其说明

扩　展　库	说　　明
php_ftp.dll	支持 FTP 函数库，可以实现客户机与服务器之间标准传送协议（FTP）
php_gd2.dll	支持图像处理函数库，支持.gif、.jpg、.png 等多种图像格式
php_imap.dll	支持 imap 电子邮件处理函数库





Note

续表

扩 展 库	说 明
php_mssql.dll	支持 MsSQL 数据库
php_msql.dll	支持 MsSQL 数据库
php_MySQL.dll	支持 MySQL 数据库
php_oracle.dll	支持 Oracle 数据库
php_pdf.dll	支持 PDF 文件处理函数库
php_sockets.dll	支持 Sockets 处理函数库
php_zlib.dll	支持 zlib 文件压缩函数库
php_pdo.dll	支持 PDO 数据库抽象层
php_pdo_mysql.dll	支持 MySQL 数据库
php_pdo_mssql.dll	支持 Ms SQL Server 数据库
php_pdo_oci8.dll	支持 Oracle 数据库
php_pdo_odbc.dll	支持 ODBC 数据库
php_pdo_pgsql.dll	支持 PGSQL 数据库

小测试

解析 AppServ 安装完成后，AppServ 文件夹下的 4 个子目录如图 1.22 所示。

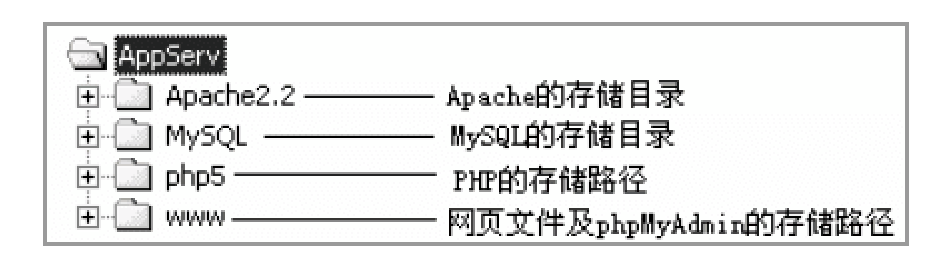


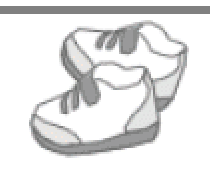
图 1.22 AppServ 的目录结构

- ☑ 在 Apache2.2\conf\目录下有一个 httpd.conf 文件，它是 Apache 服务器的配置文件，在这个文件中可以修改 Apache 服务器的端口号、根目录等，所有有关 Apache 服务器的配置都在这个文件中完成。
- ☑ 在 MySQL 目录下有一个 my.ini 文件，它是 MySQL 服务器的配置文件，存储 MySQL 的配置信息。
- ☑ 在 MySQL\data 目录下存储的是数据库文件，所有程序使用的数据库都存储在这个文件夹下。
- ☑ 在 php5\ext 文件夹下存储的是 PHP 内置的函数类库，以.dll 的格式存储。而 PHP 的配置文件 php.ini 是存储在本机系统盘的 Windows 文件夹下。
- ☑ www，程序运行的根目录，也就是说所有要运行的程序都必须存储在这个目录下。phpMyAdmin 图形化管理工具默认就存储在这个目录下。

1.4 解决 PHP 的常见配置问题

视频讲解：配套资源\mr\1\video\解决 PHP 的常见配置问题.exe

程序运行出错是许多程序员最为头疼的问题，下面介绍 PHP 常见的配置问题。通过本节的学习，可以分清哪些错误是由于 PHP 环境配置不当产生的，从而避免不必要的时间资源浪费，高效地完成 Web 应用程序的开发。



1.4.1　解决 Apache 服务器端口冲突

IIS 的默认端口号为 80，同 Apache 服务器默认端口号相同。由于采用了相同的端口号 80，因此，在运行网页时就会发生冲突。

如果用户机器上安装了 IIS，就需要修改 IIS 的默认端口，否则将导致 Apache 服务器无法正常工作。更改 IIS 的默认侦听端口 80，可以在 IIS 的管理器中进行设置，或者停止 IIS 的服务也可以。



用户也可以在安装 Apache 服务器时将默认的端口号进行更改，从而解决两个服务器共用一个端口号而产生冲突的问题。

指点迷津：

如果在搭建 PHP 环境时，将 Apache 的端口号设置为 82，那么在通过浏览器访问项目时，则应该输入 http://127.0.0.1:82/或者 http://localhost:82/。

1.4.2　设置 PHP 的系统当前时间

由于 PHP5.0 对 date()函数进行了重写，因此，目前的日期时间函数比系统时间少 8 个小时。在 PHP 语言中默认设置的是标准的格林威治时间（即采用的是零时区），所以要获取本地当前的时间必须更改 PHP 语言中的时区设置。方法如下：

在 php.ini 文件中，将[date]下的;date.timezone =选项修改为 date.timezone =Asia/Hong_Kong，然后重新启动 Apache 服务器。

设置完成后，在输出系统当前的时间就不会出现时差问题。

1.4.3　增加 PHP 扩展模块

增加 PHP 扩展模块也称为动态扩展，用来动态加载某个模块，它包含一个指令：extension。

在 Windows 操作系统下，加载模块的方法为：打开 php.ini 文件，定位到如下位置，去掉;tension=php_java.dll 前面的分号，保存后重新启动 Apache 服务器，即完成扩展模块的加载操作。

```
;tension=php_java.dll
```

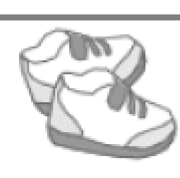
在 Linux 操作系统下，加载模块的方法如下。

```
extension=php_java.so
```

需要说明的是，只加载这一行代码并不一定能启用相关的扩展包，有时还需要确保在操作系统中安装相关的软件，例如，为启用 java 支持，需要安装 JDK。

本 章 摘 要

1．PHP 环境搭建和开发工具的选择。
2．AppServ 搭建 PHP 开发环境。
3．XAMPP 搭建 PHP 开发环境。
4．HTTP.CONF 配置 Apache 服务器。



5. PHP.INI 配置 PHP。
6. PHP 常见配置问题处理。



习　　题

1. 下面几个后台脚本编程语言中，哪个属于开源的软件（　　）。
   A. ASP　　B. PHP　　C. JSP　　D. CGI
2. 在下面哪个文件夹中能找到修改 Apache 服务器的端口号 httpd.conf 文件（　　）。
   A. conf　　B. bin　　C. error　　D. data
3. 在本书中更改 PHP 语言中的时区设置有（　　）种方法。
   A. 1　　B. 2　　C. 3　　D. 4
4. Apache 服务器默认的端口号为（　　）。
   A. 80　　B. 81　　C. 82　　D. 88
5. 下面的代码，输出的内容是（　　）。

```
<?php
echo "学习编程贵在坚持！";
?>
```

   A. 输出内容为空　　B. 学习编程贵在坚持！　　C. 学习编程贵在坚持！！
6. 使用日期时间函数之前添加（　　）不会出现时差。
7. php.ini 存储于系统盘的（　　）文件夹中。
8. 测试 AppServ 是否安装成功，方法为：在 IE 地址栏输入（　　）或（　　）。
9. AppServ 目录下有（　　）个子目录。
10. 下面是输出时间的代码，将其补充完整。

```
<?php
echo （　　　　）("Y-m-d H:i:s");
?>
```

实战模拟

实战模拟 1　编写第一个 PHP 程序

通过 PHP 脚本输出一段欢迎信息。程序代码如下：

```
<?php
echo "明日科技欢迎您！";
?>
```

其运行结果如图 1.23 所示。

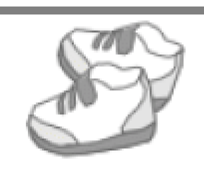
实战模拟 2　输出系统的当前时间

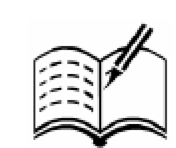
编写 PHP 脚本，应用 date()函数格式化输出系统的当前时间，其代码如下：

```
<?php
echo date("Y年 n月 j日");
?>
```

PHP 代码分析：

☑ <?php 和?>是 PHP 的标记对。在这对标记中的所有代码都被当作 PHP 代码来处理。





- ☑ echo 是 PHP 中的输出语句，与 ASP 中的 response.write、JSP 中的 out.print 含义相同，输出字符串或变量值。每行代码均以分号“;”结尾。
- ☑ date()是 PHP 中的日期时间格式化函数，在本范例中以“1900 年 1 月 1 日”格式输出系统的当前时间。

其运行效果如图 1.24 所示。

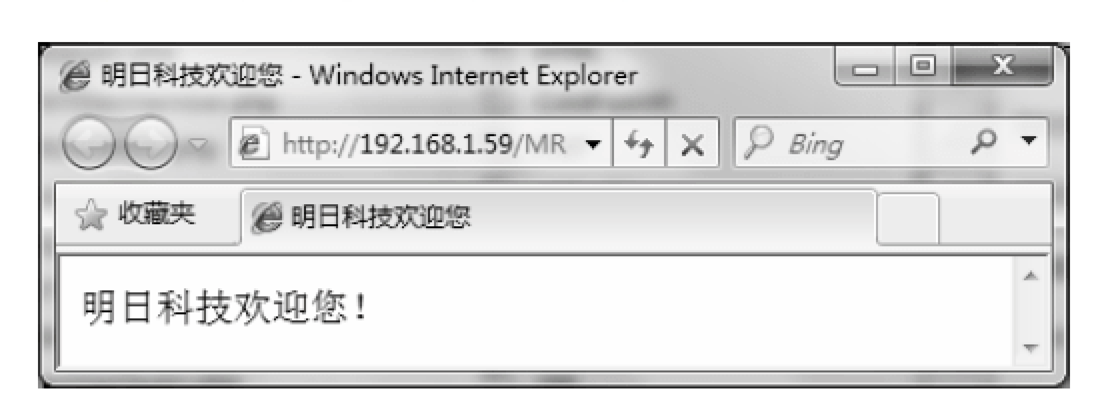




图 1.23　PHP 页面运行结果



图 1.24　输出的当前系统时间

实战模拟 3　更改 Apache 服务器的端口号为 82

当用户安装完 Apache 后再进行安装其他软件时，有时候会出现服务器端口号发生冲突，导致其他软件安装不成功的现象，例如安装迅雷。此时用户只需要将 Apache 服务器的端口号改为其他的数值即可。这里笔者将 Apache 服务器端口号改为 82。其具体做法如下：

（1）打开 Apache 目录下的 Apache2.2 子目录，找到 conf 文件夹（例如：E:\AppServ\Apache2.2\ conf）

（2）通过记事本打开 httpd.conf 文件。

（3）按 Ctrl+F 组合键搜索 80，定位到“Listen”，将 80 端口修改为 82。

（4）单击“开始”/“管理工具”/“服务”，选择 Apache2.2，单击鼠标右键，选择“重新启动”Apache 服务器。

（5）对端口号修改后的 Apache 进行测试，打开浏览器，在地址栏中输入 http://localhost:82/，如果打开如图 1.25 所示的界面，则说明服务器端口修改成功，否则失败。

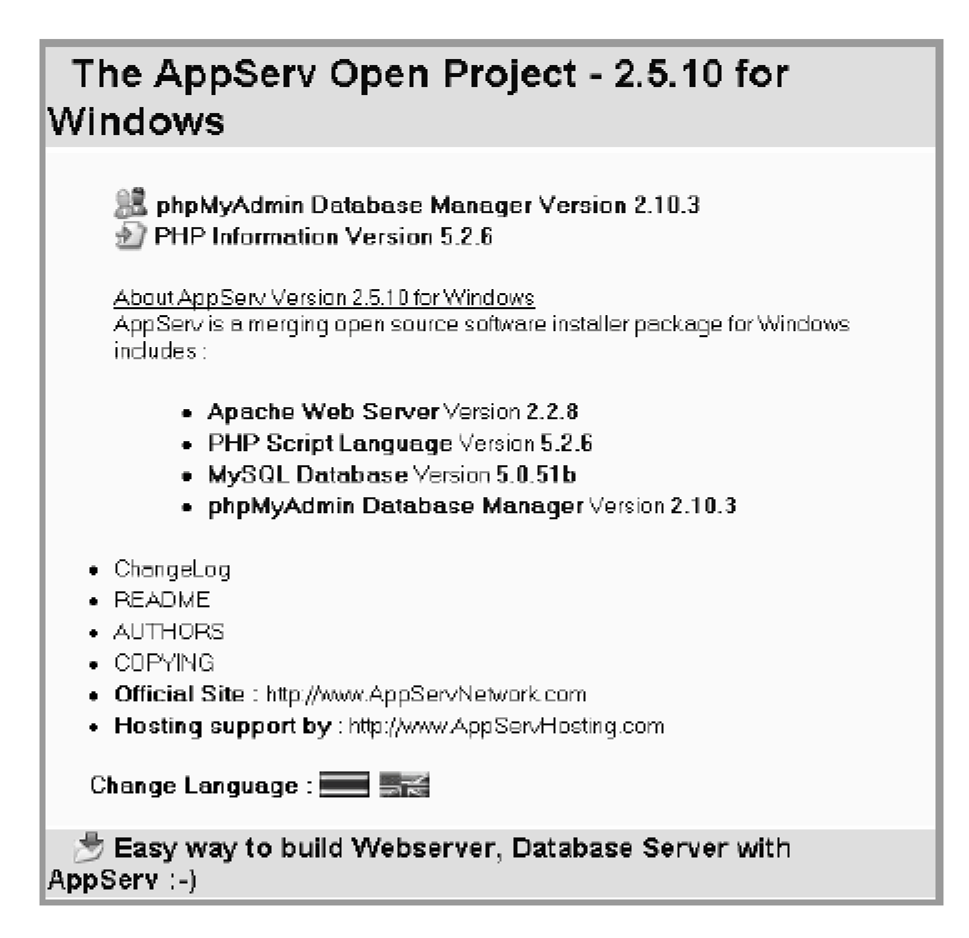


图 1.25　AppServ 测试页

指点迷津：

在 localhost 和 82 之间要用冒号（:）连接，初学者很容易忽略。

第2章

PHP 基础

（自学视频、源程序：配套资源\mr\2\）

学习一门语言，首先要学习这门语言的语法，PHP 也不例外。本章开始学习 PHP 的基础知识，也是 PHP 的核心内容。不论是网站制作，还是应用程序开发，没有扎实的基本功是行不通的。开发一个功能模块，如果一边查函数手册一边写程序代码，大概需要 15 天的时间；但基础好的人只要 3～5 天，甚至更少的时间。为了将来应用 PHP 语言开发 Web 程序节省时间，现在就要认真地从基础学起，牢牢掌握 PHP 的基础知识。只有做到这一点，才能在以后的开发过程中事半功倍。

学习摘要：

- PHP 工作原理
- PHP 标记及注释
- PHP 常量及预定义常量
- PHP 变量
- PHP 数据类型
- PHP 数据类型的转换和检测
- PHP 运算符及运算符的使用规则





Note

2.1 PHP 工作原理

PHP 即 Hypertext Preprocessor（超文本预处理器）的缩写，是基于服务器端运行的脚本程序语言，用于实现数据库和网页之间的数据交互。

一个完整的 PHP 系统由以下几部分构成。

☑ 操作系统：网站运行服务器所使用的操作系统。PHP 不要求操作系统的特定性，其跨平台的特性允许 PHP 运行在任何操作系统上，如 Windows、Linux 等。

☑ 服务器：搭建 PHP 运行环境时所选择的服务器。PHP 支持多种服务器软件，包括 Apache、IIS 等。

☑ PHP 包：实现对 PHP 文件的解析和编译。

☑ 数据库系统：实现系统中数据的存储。PHP 支持多种数据库系统，包括 MySQL、SQL Server、Oracle 及 DB2 等。

☑ 浏览器：浏览网页。由于 PHP 在发送到浏览器时已经被解析器编译成其他的代码，所以 PHP 对浏览器没有任何限制。

图 2.1 完整地展示了用户通过浏览器访问 PHP 网站系统的全过程，从图中可以更加清晰地理清它们之间的关系。

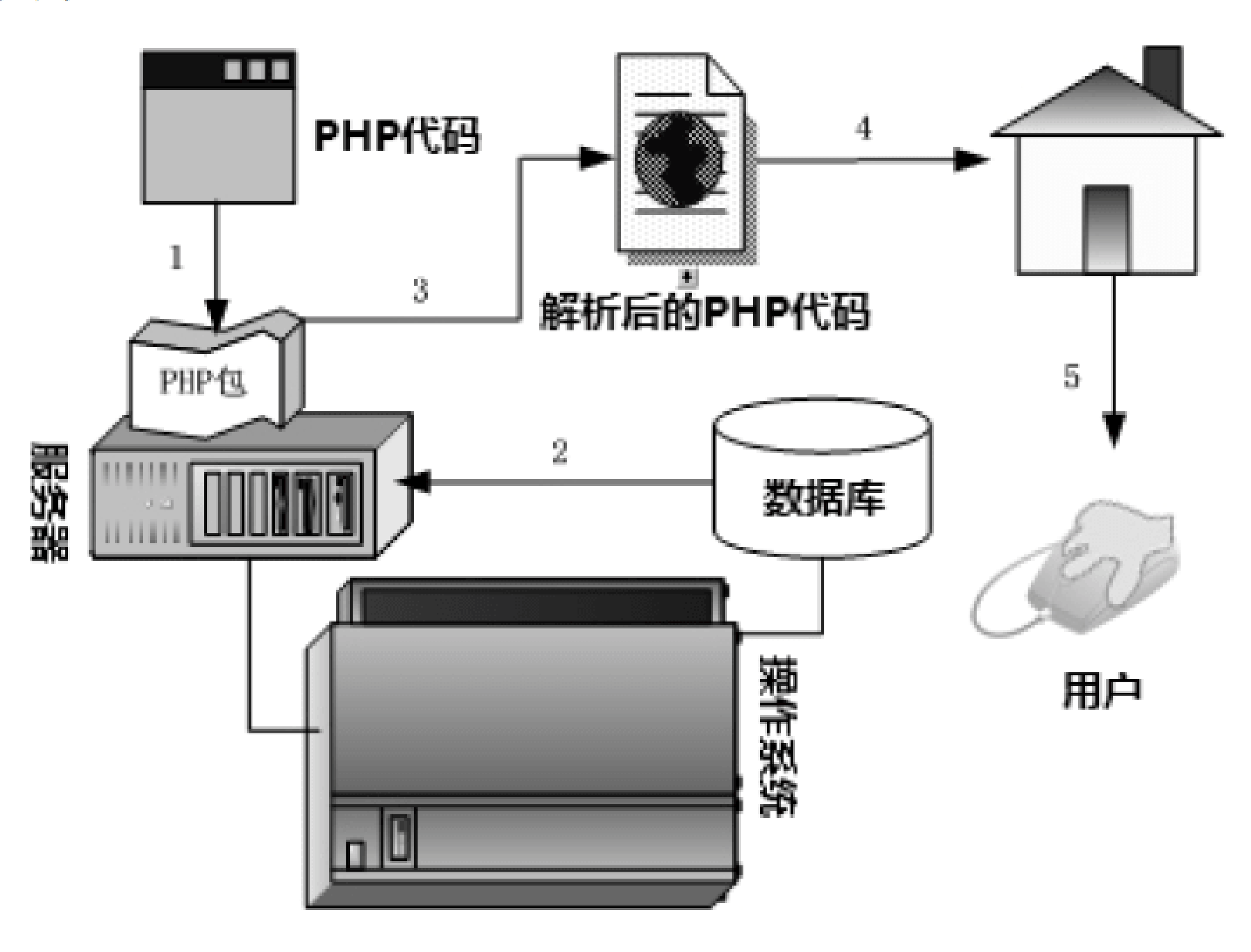


图 2.1　PHP 的工作原理

指点迷津：

PHP 的工作原理如下。

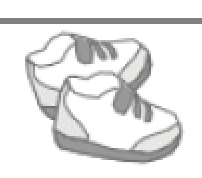
（1）PHP 的代码传递给 PHP 包，请求 PHP 包进行解析并编译。

（2）服务器根据 PHP 代码的请求读取数据库。

（3）服务器与 PHP 包共同根据数据库中的数据或其他运行变量，将 PHP 代码解析成普通的 HTML 代码。

（4）解析后的代码发送到浏览器，浏览器对代码进行分析，获取可视化内容。

（5）用户通过访问浏览器浏览网站内容。





Note

2.2　PHP 标记

视频讲解：配套资源\mr\2\video\PHP 标记.exe

所谓标记，就是为了便于与其他内容区分所使用的一种特殊标记，PHP 共支持 4 种标记风格，下面一一介绍。

1. XML 标记风格

```
<?php
echo "这是XML标记风格";
?>
```

从上面的代码中可以看到，XML 标记风格以“<?php”开始，以“?>”结尾，中间包含的代码就是 PHP 语言代码。推荐使用这种标记风格，因为它不能被服务器禁用，在 XML、XHTML 中都可以使用。

2. 脚本标记风格

```
<script language="php">
echo "这是脚本风格的标记";
</script>
```

脚本标记风格以“<script ….. ">”开始，以“</script>”结尾。

3. 简短标记风格

```
<?
echo "这是简短风格的标记";
?>
```

如果想使用这种标记风格开发 PHP 程序，则必须保证 PHP 配置文件 php.ini 中的“short_open_tag”选项值设置为“on”。

4. ASP 标记风格

```
<%
echo "这是ASP风格的标记";
%>
```

如果想使用这种标记风格开发 PHP 程序，则必须保证 PHP 配置文件 php.ini 中的“asp_tags”选项值设置为“on”。

上机演练

上机演练 1　在页面中打印 PHP 的配置信息

应用 PHP 的标记通过 echo 语句和 phpinfo()函数，向用户展示 PHP5 的配置信息，包括配置文件所在目录以及一些相关扩展库的版本、作者信息等。运行结果如图 2.2 所示。





PHP Version 5.2.6

System	Windows NT LM 5.2 build 3790
Build Date	May 2 2008 18:01:20
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared" "--with-extra-includes=C:\Program Files (x86)\Microsoft SDK\Include;C:\PROGRA~2\MICROS~2\VC98\ATL\INCLUDE;C:\PROGRA~2\MICROS~2\VC98\INCLUDE;C:\PROGRA~2\MICROS~2\VC98\MFC\INCLUDE" "--with-extra-libs=C:\Program Files (x86)\Microsoft SDK\Lib;C:\PROGRA~2\MICROS~2\VC98\LIB;C:\PROGRA~2\MICROS~2\VC98\MFC\LIB"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\WINDOWS\php.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	enabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, data, http, ftp, compress.zlib
Registered Stream Socket Transports	tcp, udp
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, zlib.*

图 2.2　PHP 配置信息

2.3　代码注释

注释可以理解为代码中的解释和说明，是程序中不可缺少的一个重要元素，使用注释不仅能够提高程序的可读性，而且还有利于程序的后期维护工作。注释不会影响程序的执行，因为注释部分的内容不会被解释器执行。

2.3.1　使用 PHP 注释

PHP 的注释有 3 种风格，下面分别进行介绍。

1. C++风格的单行注释（//）

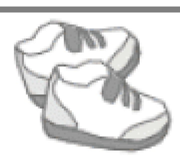
```
<?php
echo "使用C++风格的注释";
//echo "这就是C++风格的注释";
?>
```

运行结果为：使用 C++风格的注释

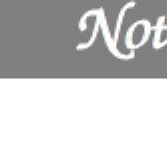
上面代码使用 echo 输出语句分别输出了“使用 C++风格的注释”和“这就是 C++风格的注释”，但是因为使用注释符号“//”将第 2 个输出语句注释掉了，所以不会被程序执行。

2. C 风格的多行注释（/*…*/）

```
<?php
    /*
```




```
        echo "这是第一行注释信息";
        echo "这是第二行注释信息";
        */
        echo "使用C风格的注释";
?>
```



Note

运行结果为：使用 C 风格的注释

上面代码虽然使用 echo 输出语句分别输出了“这是第一行注释信息”、“这是第二行注释信息”和“使用 C 风格的注释”，但是因为使用了注释符号“/*…*/”将前两个输出语句注释掉了，所以没有被程序执行。

3. Shell 风格的注释（#）

```
<?php
echo "这是Shell脚本风格的注释";      #这里的内容是看不到的
?>
```

运行结果为：这是 Shell 脚本风格的注释

因为使用了注释符号“#”，所以在“#”注释符号后面的内容是不会被程序执行的。

脚下留神：

在使用单行注释时，注释内容中不要出现“?>”标志，因为解释器会认为这是 PHP 脚本，而去执行“?>”后面的代码。例如：

```
<?php
echo"这样会出错的!!!!! "      //不会看到?>会看到
?>
```

运行结果为：这样会出错的!!!!! 会看到?>

2.3.2 有效使用注释

程序注释是书写规范程序时很重要的一个环节。注释主要是针对代码的解释和说明，用来解释脚本的用途、版权说明、版本号、生成日期、作者和内容等，有助于用户对程序的阅读和理解。合理使用注释有以下几项原则：

（1）注释语言必须准确、易懂、简洁。

多学两招：

错误的注释不但无益反而有害。

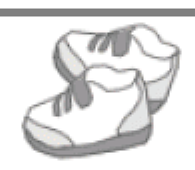
（2）注释在编译代码时会被忽略，不会被编译到最后的可执行文件中，所以注释不会增加可执行文件的大小。

（3）注释可以书写在代码中的任意位置，但是一般写在代码的开头或者结束位置。

多学两招：

避免在一行代码或表达式的中间插入注释，否则容易使代码的可理解性变差。

（4）修改程序代码时，一定要同时修改相关的注释，以保持代码和注释的同步。



（5）在实际的代码规范中，要求注释占程序代码的比例达到 20%左右，即 100 行程序中包含 20 行左右的注释。

多学两招：

防止出现没必要的重复注释信息。



（6）在程序块的结束行右方加注释标记，以表明某程序块的结束。

（7）避免在注释中使用缩写，特别是非常用缩写。

（8）注释与所描述内容进行同样的缩排，可使程序排版整齐，并方便注释的阅读与理解。

2.4　PHP 常量

视频讲解：配套资源\mr\2\video\PHP 常量.exe

常量用于存储不经常改变的数据信息。常量的值被定义后，在程序的整个执行期间内，该值都有效，并且不可再次对该常量进行赋值。

2.4.1　声明和使用常量

1. 使用 define()函数声明常量

在 PHP 中，使用 define()函数来定义常量，函数的语法如下：

```
define(string constant_name,mixed value,case_sensitive=true)
```

define()函数的参数说明如表 2.1 所示。

表 2.1　define()函数的参数说明

参　　数	说　　明
constant_name	必选参数，常量名称，即标志符
value	必选参数，常量的值
case_sensitive	可选参数，指定是否大小写敏感，设定为 True，表示不敏感

2. 使用 constant()函数获取常量的值

获取指定常量的值和直接使用常量名输出的效果是一样的，但函数可以动态地输出不同的常量，在使用上更加灵活、方便。constant()函数的语法如下：

```
mixed constant(string const_name)
```

参数 const_name 为要获取常量的名称。如果成功，则返回常量的值；如果失败，则提示错误信息常量没有被定义。

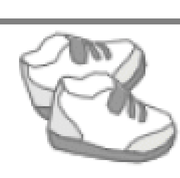
3. 使用 defined()函数判断常量是否已经被定义

defined()函数的语法如下：

```
bool defined(string constant_name);
```

参数 constant_name 为要获取常量的名称，成功则返回 True；否则，返回 False。

例 2.1　使用 define()函数来定义名为 MESSAGE 的常量，然后使用 constant()函数来获取



该常量的值，最后使用 defined()函数来判断常量是否已经被定义，代码如下：**（实例位置：配套资源\mr\2\example\2.1）**

```
<?php
/*使用define()函数来定义名为MESSAGE的常量，并为其赋值为“能看到一次”，然后分别输出常量MESSAGE和Message，因为没有设置Case_sensitive参数为true，所以表示大小写敏感，因此执行程序时，解释器会认为没有定义该常量而输出提示，并将Message作为普通字符串输出 */
define("MESSAGE","能看到一次");
echo MESSAGE;
echo Message;
/*使用define()函数来定义名为COUNT的常量，并为其赋值为“能看到多次”，并设置Case_sensitive参数为true，表示大小写不敏感，分别输出常量COUNT和Count，因为设置了大小不敏感，因此程序会认为它和COUNT是同一个常量，同样会输出值*/
define("COUNT","能看到多次",True);
echo "<br>";
echo COUNT;
echo "<br>";
echo Count;
echo "<br>";
echo constant("Count");              //使用constant()函数来获取名为Count常量的值，并输出
echo "<br>";                         //输出空行符
echo (defined("MESSAGE"));           //判断MESSAGE常量是否已被赋值，如果已被赋值，则输出“1”；如果未被赋值，则返回False
?>
```

运行结果如图 2.3 所示。

指点迷津：

在运行本实例时，由于 PHP 环境配置的不同（php.ini 中错误级别设置的不同），可能会出现不同的运行结果。图 2.3 中展示的是将 php.ini 文件中“error_reporting”的值设置为“E_ALL”后的结果。如果将“error_reporting”的值设置为“E_ALL & ~E_NOTICE”，那么将会输出如图 2.4 所示的结果。

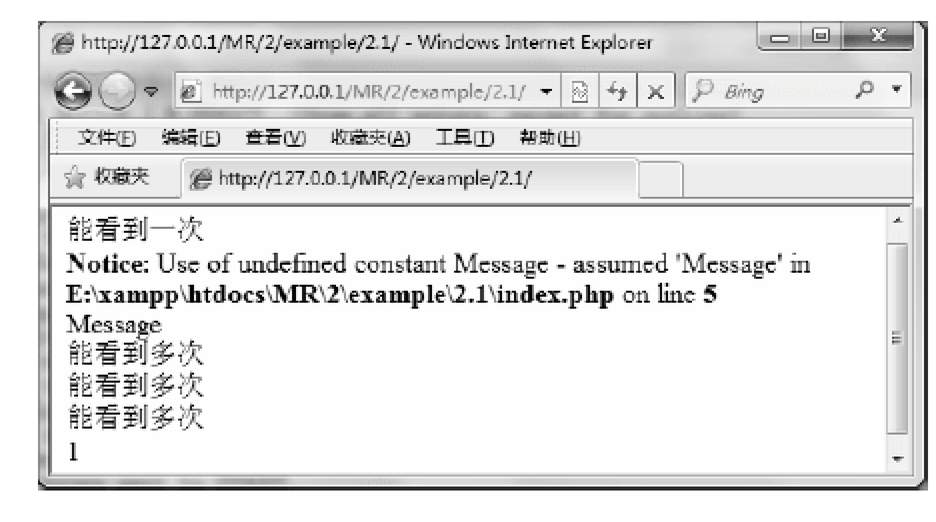


图 2.3　常量的输出结果

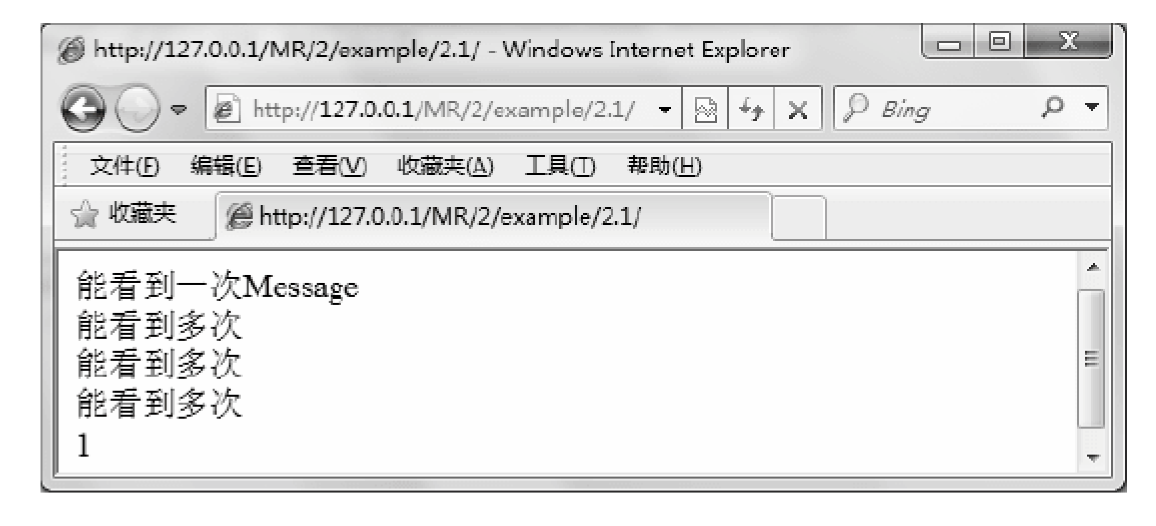


图 2.4　常量的输出结果

2.4.2　预定义常量

PHP 中提供了很多预定义常量，可以获取 PHP 中的信息，但不能任意更改这些常量的值。PHP 中预定义常量的名称及其作用如表 2.2 所示。

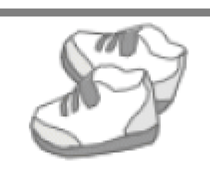


表 2.2　PHP 中预定义常量的名称及其作用

常　量　名	作　　用
__FILE__	默认常量，PHP 程序文件名
__LINE__	默认常量，PHP 程序行数
PHP_VERSION	内建常量，PHP 程序的版本，如“3.0.8_dev”
PHP_OS	内建常量，执行 PHP 解析器的操作系统名称，如“Windows”
TRUE	该常量是一个真值（True）
FALSE	该常量是一个假值（False）
NULL	一个 null 值
E_ERROR	该常量指到最近的错误处
E_WARNING	该常量指到最近的警告处
E_PARSE	该常量指到解析语法有潜在问题处
E_NOTICR	该常量为发生不寻常，但不一定是错误处

多学两招：

__FILE__和__LINE__中的“__”是两条下划线，而不是一条“_”。表 2.2 中以 E_开头的预定义常量是 PHP 的错误调试部分。若想详细了解，可参考 error_reporting()函数。

例 2.2　使用预定义常量来输出 PHP 中的一些信息，代码如下：（**实例位置：配套资源\mr\2\example\2.2**）

```
<?php
echo "当前文件路径为：".__FILE__;                //使用__FILE__常量获取当前文件路径
echo "<br>";
echo "当前行数为：".__LINE__;                    //使用__LINE__常量获取当前所在行数
echo "<br>";
echo "当前PHP版本信息为：".PHP_VERSION;          //使用PHP_VERSION常量获取当前PHP版本
echo "<br>";
echo "当前操作系统为：".PHP_OS;                  //使用PHP_OS常量获取当前操作系统
?>
```

运行结果如图 2.5 所示。

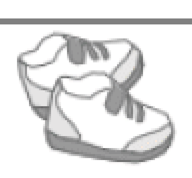


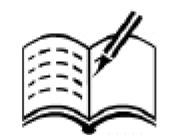
图 2.5　使用预定义常量获取 PHP 信息

上机演练

上机演练 2　获取当前执行文件的名称

通过__FILE__预定义常量获取目标文件的路径及名称，运行结果如图 2.6 所示。





Note

上机演练 3　计算一个圆形的面积

常量是 PHP 编程基础的重要组成部分，其作用是定义一个不会改变的值。本练习通过计算圆的面积向用户说明常量是如何定义和使用的，运行结果如图 2.7 所示。

D:\AppServ\www\mr\2\paradigm\02\index.php

图 2.6　输出当前文件的路径和名称

半径为12个单位的圆的面积452.3893344

图 2.7　使用常量指定 PI 的值计算圆的面积

2.5　PHP 变量

视频讲解：配套资源\mr\2\video\PHP 变量.exe

变量是指可以随时改变的量，主要用于存储临时数据信息，是编写程序中尤为重要的一部分。在定义变量时，通常要为其赋值，所以在定义变量的同时，系统会自动为该变量分配一个存储空间来存放变量的值。

2.5.1　声明变量

1. 变量的定义

在 PHP 中，变量的语法格式如下：

```
$变量名称=变量的值
```

2. 变量的命名规则

变量的命名规则包括：

☑ 在 PHP 中的变量名是区分大小写的。
☑ 变量名必须以美元符号（$）开始。
☑ 变量名可以以下划线开始。
☑ 变量名不能以数字字符开始。
☑ 变量名可以包含一些扩展字符（如重音拉丁字母），但不能包含非法扩展字符（如汉字字符和汉字字母）。

正确的变量命名：

```
$name="mingri";           //定义一个变量，变量名为$name，变量值为mingri
$_pwd="roof";             //定义一个变量，变量名为$_pwd，变量值为roof
$_123number=87665;        //定义一个变量，变量名为$_123number，变量值为87665
$_Class="roof";           //定义一个变量，变量名为$_Class，变量值为roof
```

错误的变量命名：

```
$11112_var=11112;         //变量名不能以数字字符开头
$~%$_var="Lit";           //变量名不能包含非法字符
```

2.5.2　变量赋值

变量的赋值有直接赋值、传值赋值和引用赋值 3 种方式。



Note

1. 直接赋值

直接赋值就是使用“=”直接将值赋给某变量，例如：

```
<?php
$name=mingri;
$number=30;
echo $name;
echo $number;
?>
```

运行结果为：

```
mingri
30
```

上例中分别定义了$name 变量和$number 变量，并分别为其赋值，然后使用 echo 输出语句输出变量的值。

2. 传值赋值

传值赋值就是使用“=”将一个变量的值赋给另一个变量，例如：

```
<?php
$a=10;
$b=$a;
echo $a."<br>";
echo $b;
?>
```

运行结果为：

```
10
10
```

在上面的例子中，首先定义变量 a 并将其赋值为 10，然后定义变量 b，并设置变量 b 的值等于变量 a 的值，此时变量 b 的值也为 10。

3. 引用赋值

引用赋值是指一个变量引用另一个变量的值，例如：

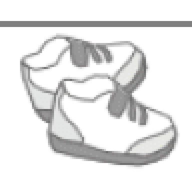
```
<?php
$a=10;
$b=&$a;
$b=20;
echo $a."<br>";
echo $b;
?>
```

运行结果为：

```
20
20
```

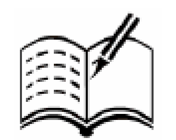
仔细观察一下，“$b=&$a”中多了一个“&”符号，这就是引用赋值。当执行“$b=&$a”语句时，变量 b 将指向变量 a，并且和变量 a 共用同一个值。

当执行“$b=20”时，变量 b 的值发生了变化，此时，由于变量 a 和变量 b 共用同一个值，



所以当变量 b 的值发生变化时，变量 a 也将随之发生变化。

2.5.3 变量作用域



变量的作用域是指变量在哪些范围内能被使用，在哪些范围内不能被使用。PHP 中分为 3 种变量作用域，分别为局部变量、全局变量和静态变量。

1. 局部变量

局部变量就是在函数的内部定义的变量，其作用域是所在函数。

例 2.3 自定义一个名为 example()的函数，然后分别在该函数内部及函数外部定义并输出变量 a 的值，具体代码如下：（**实例位置：配套资源\mr\2\example\2.3**）

```
<?php
function example(){
    $a="hello php!";          //在自定义函数example()中定义变量a
    echo "在函数内部定义的变量a的值为：".$a."<br>";
}
example();
$a="hello china!";            //在函数外部定义变量a
echo "在函数外部定义的变量a的值为：".$a."<br>";
?>
```

运行结果如图 2.8 所示。

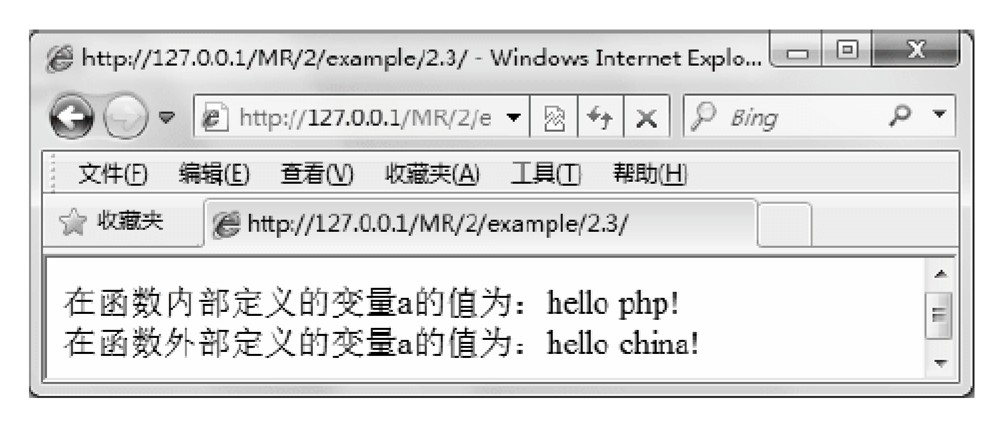


图 2.8 局部变量示例运行结果图

指点迷津：

在例 2.3 中，只要了解局部变量的使用方法及意义即可，函数相关内容将在第 3 章中详细讲解。

2. 全局变量

全局变量是被定义在所有函数以外的变量，其作用域是整个 PHP 文件，但是在用户自定义函数内部是不可用的。想在用户自定义函数内部使用全局变量，要使用 global 关键词声明。

例 2.4 定义一个全局变量，并且在函数内部输出全局变量的值，具体代码如下：（**实例位置：配套资源\mr\2\example\2.4**）

```
<?php
$a="hello php!";              //在自定义函数外部声明一个变量a
function example(){           //自定义一个函数，名为example
    global $a;                //使用global关键词声明并使用在函数外部定义的变量a
    echo "在函数内部获得变量a的值为：".$a."<br>";
}
```



```
    example();
?>
```

运行结果如图 2.9 所示。

3. 静态变量

通过局部变量的定义可以知道，在函数内部定义的变量，在函数调用结束后，其变量将会失效。但有时仍然需要该函数内的变量有效，此时就需要将变量声明为静态变量。声明静态变量只需在变量前加 static 关键字即可。

例 2.5 分别在函数内声明静态变量和局部变量，并且执行函数，比较执行结果有什么不同。具体代码如下：（**实例位置：配套资源\mr\2\example\2.5**）

```
<?php
function example(){
    static $a=10;         //定义静态变量
    $a+=1;
    echo "静态变量a的值为：".$a."<br>";
}
function xy(){
    $b=10;                //定义局部变量
    $b+=1;
    echo "局部变量b的值为：".$b."<br>";
}
example();                //一次执行该函数体
example();                //二次执行该函数体
example();                //三次执行该函数体
xy();                     //一次执行该函数体
xy();                     //二次执行该函数体
xy();                     //三次执行该函数体
?>
```

运行结果如图 2.10 所示。

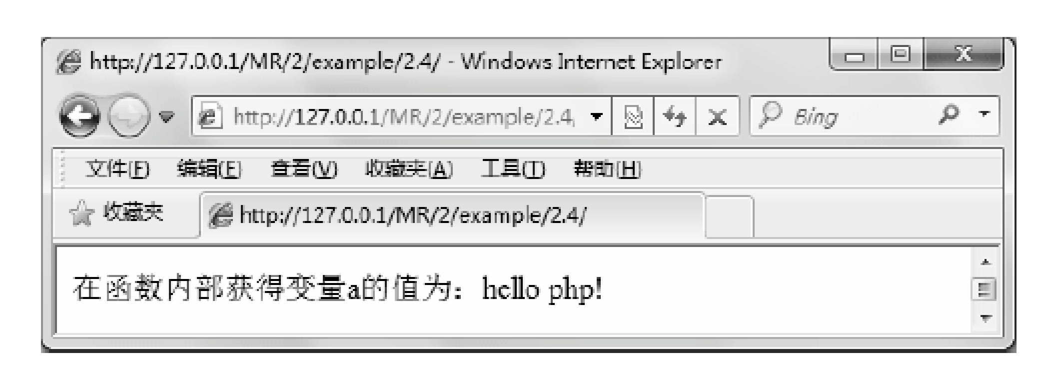


图 2.9 全局变量示例运行结果

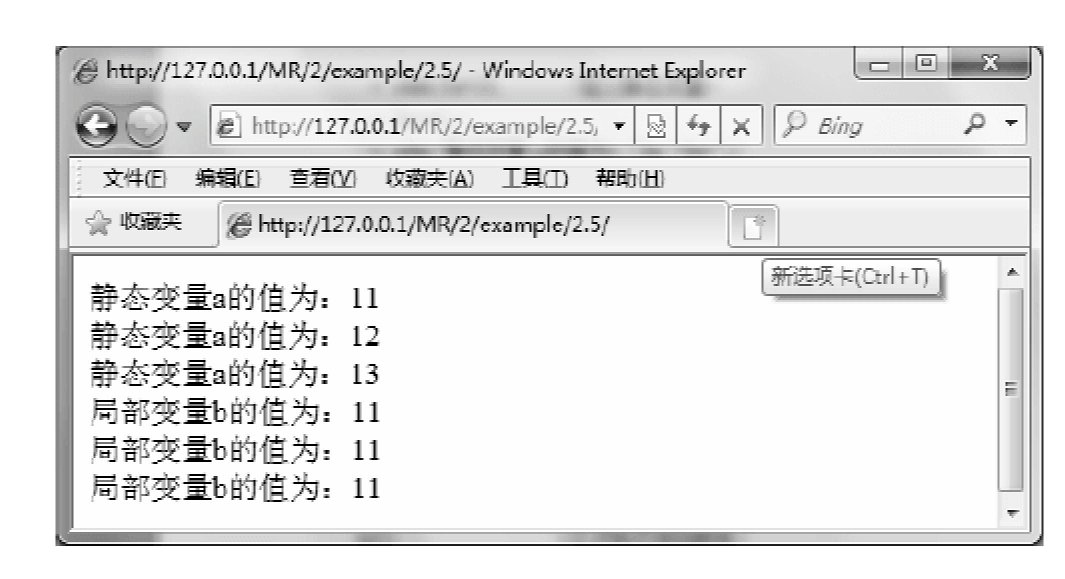


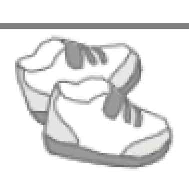
图 2.10 静态变量与局部变量的区别

2.5.4 可变变量

可变变量是一种独特的变量，这种变量的名称是由另外一个变量的值来确定的。声明可变变量的方法是在变量名称前加两个“$”符号。

声明可变变量的语法如下：

```
$$可变变量名称=可变变量的值
```





例 2.6 举例说明声明可变变量的方法，具体代码如下：（**实例位置：配套资源\mr\2\example\2.6**）

```
<?php
$a="mrkj";                  //定义变量
$$a="bccd";                 //声明可变变量，该变量名称为变量a的值
echo $a."<br>";             //输出变量a
echo $$a."<br>";            //输出可变变量
echo $mrkj;                 //输出变量mrkj
?>
```

运行结果如图 2.11 所示。

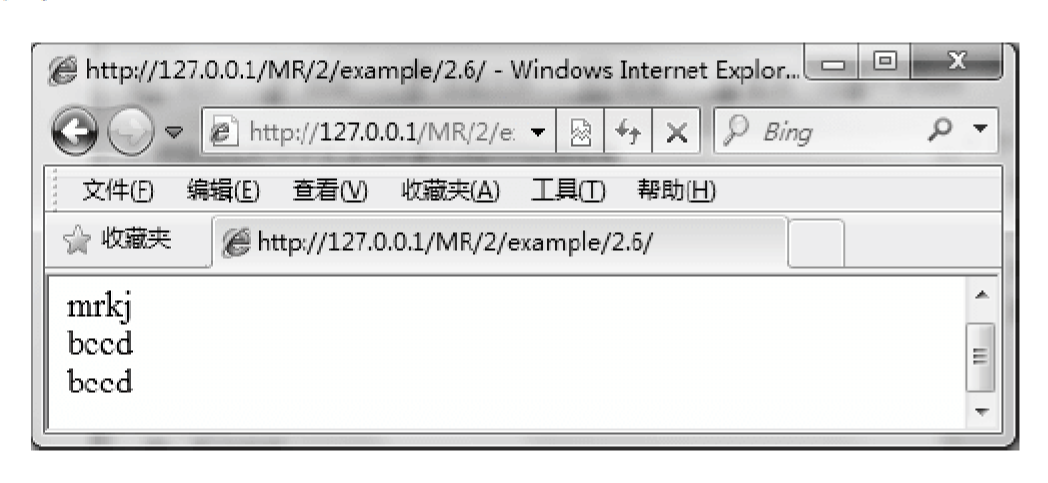


图 2.11　可变变量示例运行结果

☏小测试

制作一个简单的加法计算器，当单击“等于”按钮时，通过算术运算符“+”计算两个文本框中数据的和，并显示在后面的文本框中，运行结果如图 2.12 所示。

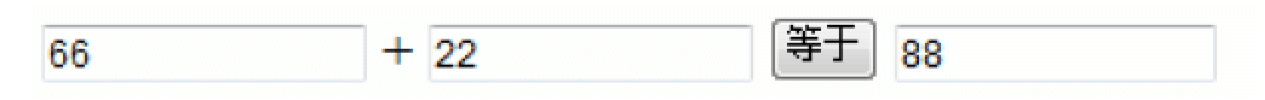


图 2.12　加法计算器

具体步骤如下：

（1）创建 index.php 文件，设计表单页面，效果如图 2.12 所示。

（2）将表单中的数据提交到本页，并通过$_POST 获取表单中提交的数据，从而完成加法运算，代码如下：

```
<?php
if(isset($_POST['Submit']) and $_POST['number1']!=null and $_POST['number2']!=null){
    $number3=$_POST['number1']+$_POST['number2'];                //计算变量值
}else{
    $number3=null;
}
?>
```

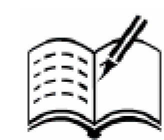
2.6　PHP 数据类型

视频讲解：配套资源\mr\2\video\PHP 数据类型.exe

在计算机的世界中，计算机操作的对象是数据，而每一个数据都有其类型，只有具备相同类型的数据才可以彼此操作。

PHP 的数据类型可以分成 3 种，即标量数据类型、复合数据类型和特殊数据类型。





2.6.1 标量数据类型

标量数据类型是数据结构中最基本的单元，其只能存储一个数据。PHP 中的标量数据类型又可细分为 4 种类型，如表 2.3 所示。

表 2.3 标量数据类型

类 型	说 明
boolean（布尔型）	这是最简单的类型。只有两个值，即真（True）和假（False）
string（字符串型）	字符串就是连续的字符序列，可以是计算机所能表示的一切字符的集合
integer（整型）	整型数据类型只能包含整数，可以是正数或负数
float（浮点型）	浮点数据类型用来存储数字，和整型不同的是它有小数位

下面对各个数据类型进行详细介绍。

1. 布尔型（boolean）

布尔型是 PHP 中较为常用的数据类型之一，它保存一个真值（True）或假值（False）。布尔型数据的用法如下：

```
<?php
$a=TRUE;
$c=FALSE;
?>
```

2. 字符串型（string）

字符串是连续的字符序列，由数字、字母和符号组成。字符串中的每个字符只占用一个字节。字符包含以下几种类型。

☑ 数字类型。如 1、2、3 等。

☑ 字母类型。如 a、b、c、d 等。

☑ 特殊字符。如#、$、%、^、&等。

☑ 不可见字符。如\n（换行符）、\r（回车符）、\t（Tab 字符）等。

其中，不可见字符是比较特殊的一组字符，用来控制字符串格式化输出，在浏览器上不可见，只能看到字符串输出的结果。

例 2.7 运用 PHP 的不可见字符串完成字符串的格式输出，程序代码如下：**（实例位置：配套资源\mr\2\example\2.7）**

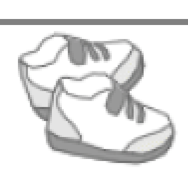
```
<?php
echo  "PHP从入门到精通\rASP从入门到精通\nJSP程序开发范例宝典\tPHP函数参考大全";
//输出字符串
?>
```

指点迷津：

\r：回车

\n：换行

\t：水平制表符



运行结果如图 2.13 所示，在 IE 浏览器中不能直接看到不可见字符串（\r、\n 和\t）的作用效果。

只有通过查看源文件才能看到不可见字符串的作用效果，如图 2.14 所示。



Note

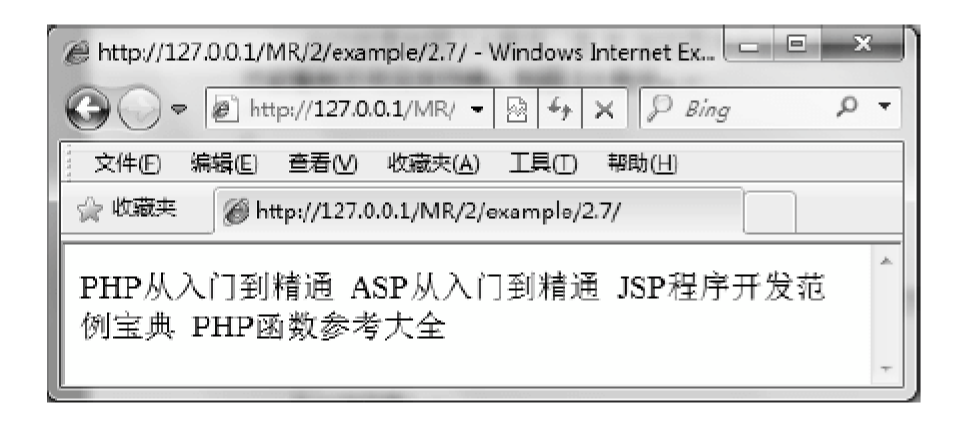


图 2.13　不可见字符串的应用

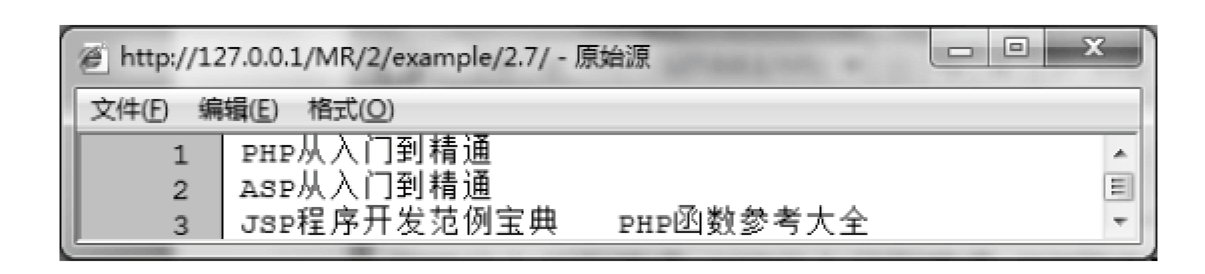


图 2.14　查看不可见字符串的作用效果

在 PHP 中，定义字符串有 3 种方式。

☑ 单引号（'）。

☑ 双引号（"）。

☑ 界定符（<<<）。

单引号和双引号是经常使用的定义方式，定义格式如下：

```
$a ='字符串';
```

或

```
$a ="字符串";
```

多学两招：

双引号中所包含的变量会自动被替换成实际数值，而在单引号中包含的变量则按普通字符串输出。

在定义字符串时，应尽量使用单引号，因为单引号的运行速度要比双引号快。

例 2.8　分别使用单引号、双引号、界定符输出变量的值，具体代码如下：（**实例位置：配套资源\mr\2\example\2.8**）

```
<?php
$a="你好！";
echo "$a"."<br>";                //使用双引号输出变量
echo '$a'."<br>";                //使用单引号输出$a
echo <<<std                      //使用界定符输出变量
    $a
std;
?>
```

运行结果如图 2.15 所示。

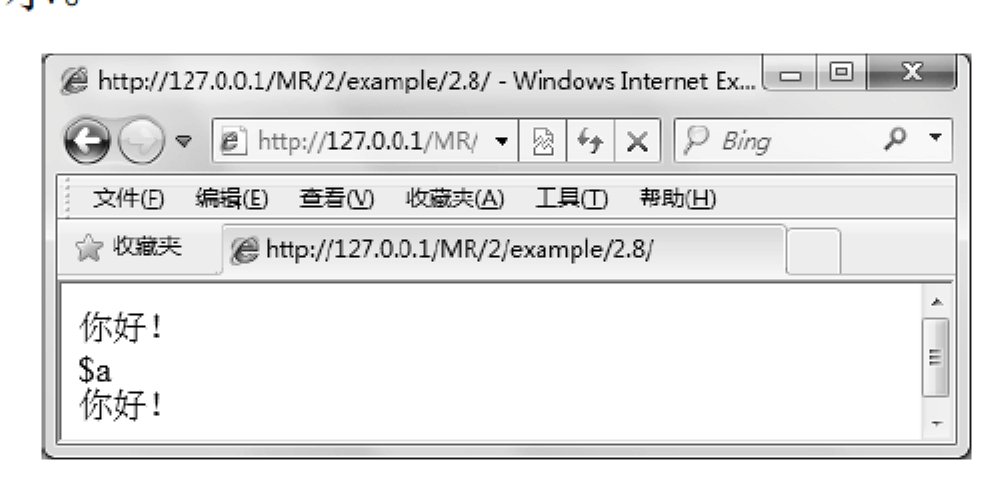
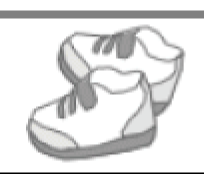
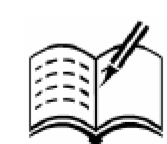


图 2.15　使用不同的方式输出变量的区别





脚下留神：

当使用界定符输出字符串时，结束标识符必须单独另起一行，并且不允许有空格。如果在标识符前后有其他符号或字符，则会发生错误。

3. 整型（integer）

整型数据类型只能包含整数。在 32 位的操作系统中，有效的范围是−2147483648～+2147483647。整型数可以用十进制、八进制和十六进制来表示。如果用八进制表示，则数字前面必须加 0；如果用十六进制表示，则需要加 0x。

例 2.9　看一个实例，分别输出八进制、十进制和十六进制的结果，具体代码如下：（**实例位置：配套资源\mr\2\example\2.9**）

```
<?php
    $str1 = 1234;                    //八进制变量
    $str2 = 01234;                   //十进制变量
    $str3 = 0x1234;                  //十六进制变量
    echo "数字1234不同进制的输出结果：<p>";
    echo "十进制的结果是：$str1<br>";
    echo "八进制的结果是：$str2<br>";
    echo "十六进制的结果是：$str3";
?>
```

运行结果如图 2.16 所示。

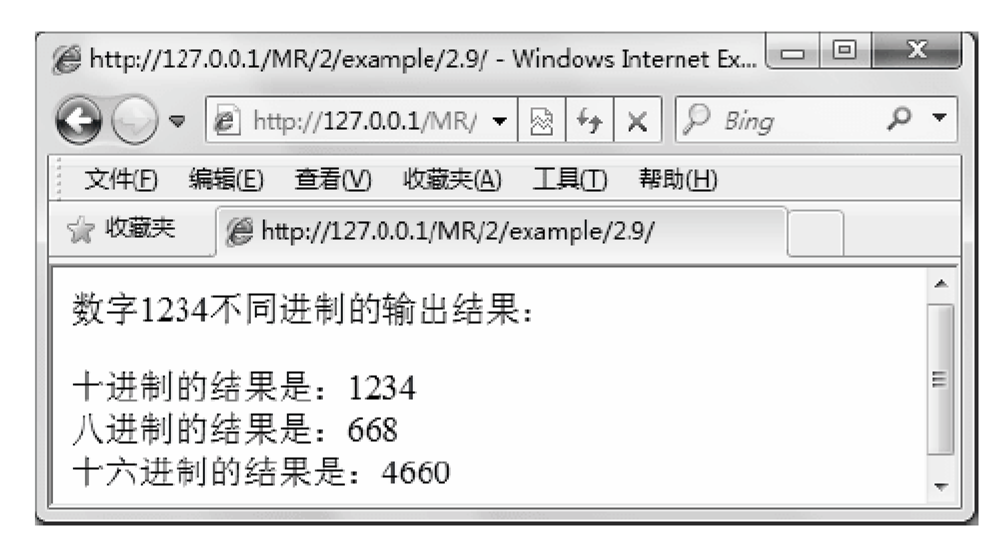


图 2.16　输出八进制、十进制和十六进制数据

脚下留神：

如果给定的数值超出了 int 类型所能表示的最大范围，将会被当作 float 型处理，这种情况称为整数溢出。同样，如果表达式的最后运算结果超出了 int 的范围，也会返回 float 型。

如果在 64 位的操作系统中，其运行结果可能会有所不同。

4. 浮点型（float）

浮点数据类型可以用来存储整数，也可以用来保存小数。其提供的精度比整数大得多。在 32 位的操作系统中，有效的范围是 1.7E−308～1.7E+308。在 PHP 4.0 以前的版本中，浮点型的标识为 double，也叫双精度浮点数，两者没什么区别。

浮点型数据默认有两种书写格式，一种是标准格式。

```
3.1415
0.333
-35.8
```





还有一种是科学记数法格式。

```
3.58E1
849.72E-3
```

例如：

```
<?php
$a=1.036;
$b=2.035;
$c=3.58E1;            //该变量的值为3.58*1²
?>
```

脚下留神：

浮点型的数值只是一个近似值，所以要尽量避免在浮点型之间比较大小，因为最后的结果往往是不准确的。

2.6.2　复合数据类型

复合数据类型包括两种：数组（array）和对象（object）。

1．数组

数组是一组数据的集合，它把一系列数据组织起来，形成一个可操作的整体。数组中可以包括很多数据，如标量数据、数组、对象、资源以及 PHP 中支持的其他语法结构等。

数组中的每个数据称为一个元素，每个元素都有一个唯一的编号，称为索引。元素的索引只能由数字或字符串组成。元素的值可以是多种数据类型。定义数组的语法格式如下：

```
$array['key'] = 'value';
```

或

```
$array(key1 => value1, key2 => value2……)
```

其中，参数 key 是数组元素的索引，value 是数组元素的值。

例 2.10　举一个简单的数组应用示例，具体代码如下：（**实例位置：配套资源\mr\2\example\2.10**）

```
<?php
$array[0]="明日科技";                              //定义$array数组的第1个元素
$array[1]="编程词典";                              //定义$array数组的第2个元素
$array[2]="编程无忧";                              //定义$array数组的第3个元素
$number=array(0=>'明日科技',1=>'编程词典',2=>'编程无忧');   //定义$number数组的所有元素
echo $array[0]."<br>";                             //输出$array数组的第1个元素值
echo $number[1];                                   //输出$number数组的第2个元素值
?>
```

运行结果如图 2.17 所示。

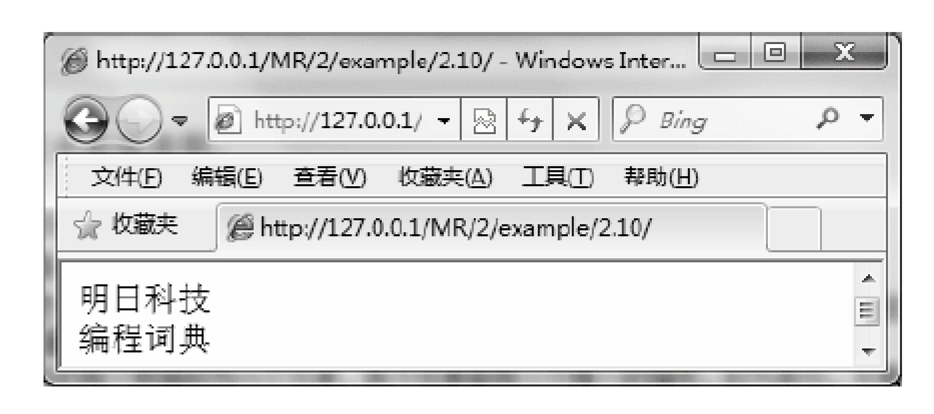
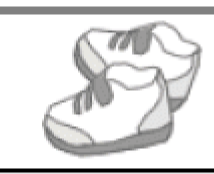
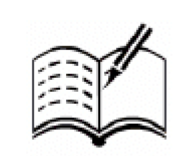


图 2.17　数组应用







指点迷津：

在本书第 9 章中将对数组进行详细讲解，此处读者只需了解什么是数组即可。

2. 对象

目前的编程语言用到的方法有两种：面向过程和面向对象。在 PHP 中，用户可以自由使用这两种方法。有关面向对象的技术可以参考本书后面的内容。

2.6.3　特殊数据类型

特殊数据类型包括两种：资源（resource）和空值（null）。

1. 资源

资源是由专门的函数来建立和使用的。它是一种特殊的数据类型，由程序员分配。在使用资源时，要及时地释放不需要的资源，如果程序员忘记了释放资源，则系统将自动启用垃圾回收机制，以避免内存消耗殆尽。

2. 空值

空值，顾名思义，表示没有为该变量设置任何值，另外，空值不区分大小写，如 null 和 NULL 的效果是一样的。被赋予空值的情况有以下 3 种。

☑ 没有赋任何值。
☑ 被赋值为 null。
☑ 被 unset()函数处理过的变量。

下面分别对这 3 种情况举例说明，具体代码如下：

```
<?php
$a;                //没有赋值的变量
$b=NULL;           //被赋空值的变量
$c=3;
unset($c);         //使用unset()函数处理后，$c的值为空
?>
```

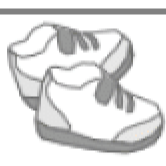
2.6.4　转换数据类型

PHP 中的类型转换和 C 语言中一样，非常简单，只需在变量前面加上一个小括号，并把目标数据类型写在小括号中即可。

PHP 中允许转换的类型如表 2.4 所示。

表 2.4　类型强制转换

转换函数	转换类型	举　例
(boolean),(bool)	将其他数据类型强制转换成布尔型	$a=1; $b=(boolean)$a; $b=(bool)$a;
(string)	将其他数据类型强制转换成字符串型	$a=1; $b=(string)$a;
(integer),(int)	将其他数据类型强制转换成整型	$a=1; $b=(int)$a; $b=(integer)$a;
(float),(double),(real)	将其他数据类型强制转换成浮点型	$a=1; $b=(float)$a; $b=(double)$a; $b=(real)$a;
(array)	将其他数据类型强制转换成数组	$a=1; $b=(array)$a;
(object)	将其他数据类型强制转换成对象	$a=1; $b=(object)$a;







在进行类型转换的过程中应该注意以下几点：

（1）转换成 boolean 型

null、0 和未赋值的变量或数组，会被转换为 False，其他转换为 True。

（2）转换成整型

布尔型的 False 转换为 0，True 转换为 1。

浮点型的小数部分被舍去。

字符串型如果以数字开头，就截取到非数字位，否则输出 0。

当字符串转换为整型或浮点型时，如果字符是以数字开头的，则会先把数字部分转换为整型，再舍去后面的字串；如果数字中含有小数点，则会取到小数点前一位。

2.6.5 检测数据类型

PHP 中提供了很多检测数据类型的函数，可以对不同类型的数据进行检测，以判断其是否属于某个类型。检测数据类型的函数如表 2.5 所示。

表 2.5 检测数据类型的函数

函　数	检 测 类 型	举　例
is_bool	检查变量是否为布尔类型	is_book($a);
is_string	检查变量是否为字符串类型	is_string($a);
is_float/is_double	检查变量是否为浮点类型	is_float($a); is_double($a);
is_integer/is_int	检查变量是否为整数	is_integer($a); is_int($a);
is_null	检查变量是否为 null	is_null($a);
is_array	检查变量是否为数组类型	is_array($a);
is_object	检查变量是否为一个对象类型	is_object($a);
is_numeric	检查变量是否为数字或由数字组成的字符串	is_numeric($a);

例 2.11 通过几个检测数据类型的函数来检测相应的字符串类型，具体代码如下：（**实例位置：配套资源\mr\2\example\2.11**）

```
<?php
$a=true;
$b="你好PHP";
$c=123456;
echo "1. 变量是否为布尔型：".is_bool($a)."<br>";          //检测变量是否为布尔型
echo "2. 变量是否为字符串型：".is_string($b)."<br>";      //检测变量是否为字符串型
echo "3. 变量是否为整型：".is_int($c)."<br>";             //检测变量是否为整型
echo "4. 变量是否为浮点型：".is_float($c)."<br>";         //检测变量是否为浮点型
?>
```

运行结果如图 2.18 所示。

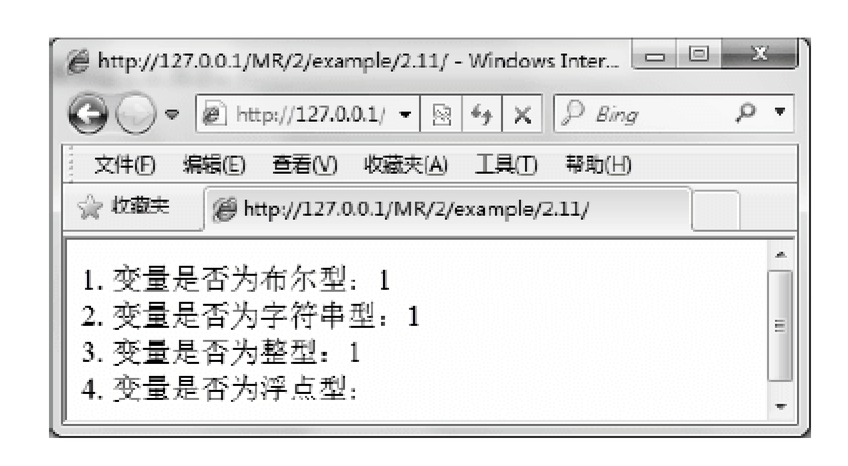
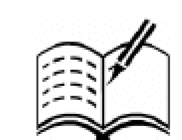


图 2.18 检测变量数据类型





指点迷津：

由于变量 C 不是浮点型，所以第 4 个判断的返回值为 False，即空值。

上机演练

上机演练 4　当数字遇到字符串

在 PHP 手册中经常会看到一些返回值为 bool 的函数，意思是当符合条件时，返回值为 1，否则返回值为 0。按照我们正常的思维返回值应该是 True 和 False，因为它们才是真正的 bool 类型。其实这些都是类型转换的结果。在 PHP 中完成数据类型转换，其运行结果如图 2.19 所示。

上机演练 5　通过转义字符输出特殊字符串

在定义长字符串变量时，往往字符串本身包含一些特殊字符。例如字符串本身包含双引号（“”），这时就需要转义字符对特殊字符进行转义。本范例通过转义字符“\”转义特殊字符双引号（“”）包含的字符串，运行结果如图 2.20 所示。

自动类型转换：
20+我是字符串型数据=20
强制类型转换：
20+我是字符串型数据=20

图 2.19　字符串类型转换后的输出结果



图 2.20　使用转义字符输出特殊字符

上机演练 6　通过 PHP5 新型字符串动态输出 JavaScript 脚本

JavaScript 语言是一门功能强大的客户端脚本语言，也是一门跨平台语言。PHP 支持使用 JavaScript 编码。通过 PHP5 新型字符串动态输出 JavaScript 代码，运行结果如图 2.21 所示。



图 2.21　动态输出 JavaScript 代码

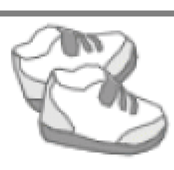
2.7　PHP 的运算符

视频讲解：配套资源\mr\2\video\PHP 的运算符.exe

运算符是用来对变量、常量或数据进行计算的符号，它对一个值或一组值执行一个指定的操作。PHP 运算符包括算术运算符、字符串运算符、赋值运算符、位运算符、递增或递减运算符、逻辑运算符、比较运算符以及三元运算符等。下面分别对各种运算符进行介绍。

2.7.1　算术运算符

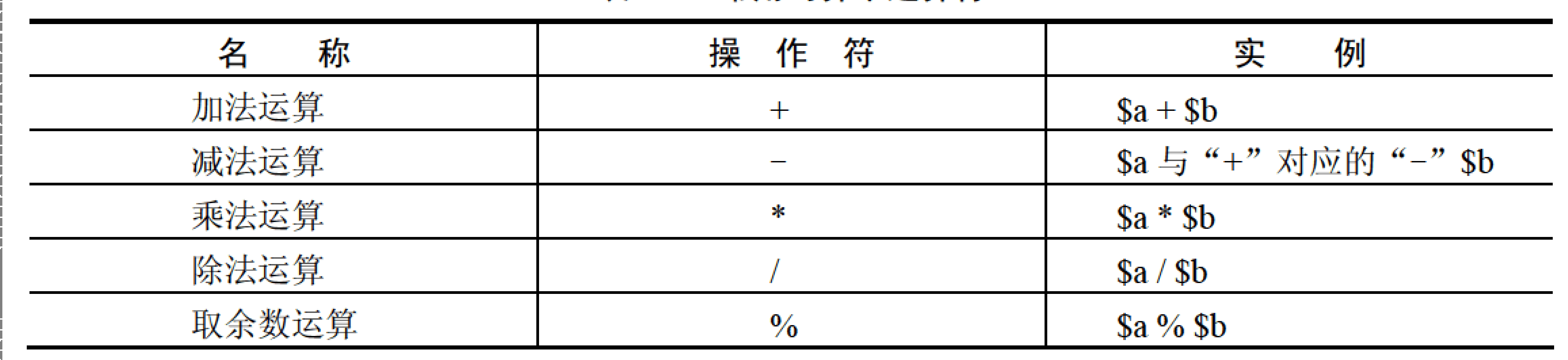
算术运算符主要用于处理算术运算操作，常用的算术运算符如表 2.6 所示。





Note

表 2.6　常用的算术运算符



名　称	操　作　符	实　例
加法运算	+	$a + $b
减法运算	−	$a 与“+”对应的“−”$b
乘法运算	*	$a * $b
除法运算	/	$a / $b
取余数运算	%	$a % $b

脚下留神：

当在算术运算符中使用“%”求余时，如果被除数（$a）是负数，那么取得的结果也是一个负值。

例 2.12　通过算术运算符计算每月总的支出、剩余工资、房贷占工资的比例等。具体代码如下：（**实例位置：配套资源\mr\2\example\2.12**）

```
<?php
$a='4000';                              //定义变量a，月工资为4000
$b='1750';                              //定义变量b，房贷为1750
$c='500';                               //定义变量b，消费金额为500
echo $c + $b .'<br>';                   //计算每月总的支出金额
echo $a-$b-$c.'<br>';                   //计算每月剩余工资
echo $b/$a.'<br>';                      //计算房贷占总工资的比例
echo $b%$a.'<br>';                      //计算变量b和变量a余数
?>
```

运行结果如图 2.22 所示。

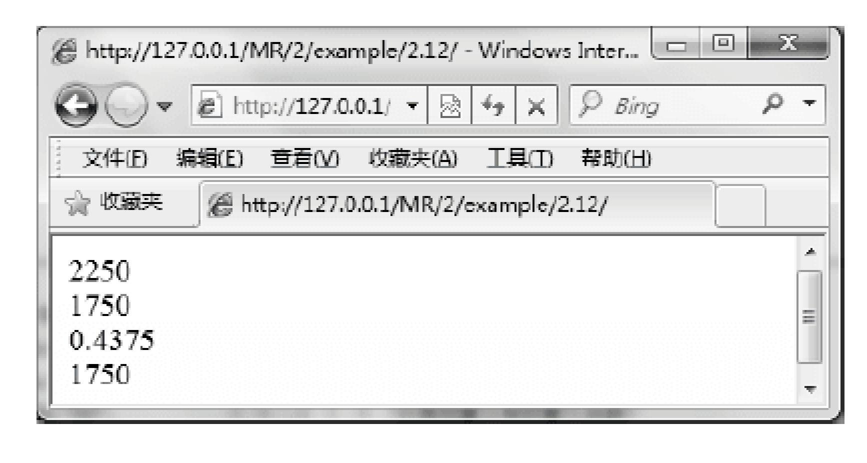


图 2.22　算术运算符示例运行结果

2.7.2　字符串运算符

字符串运算符主要用于处理字符串的相关操作，在 PHP 中字符串运算符只有一个，那就是“.”。该运算符用于将两个字符串连接起来，结合到一起形成一个新的字符串。应用格式如下：

```
$a.$b
```

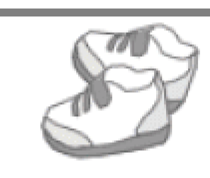
此运算符在前面的例子中已经使用，例如例 2.12 中的：

```
echo $c + $b .'<br>';                   //计算每月总的支出金额
```

此处使用字符串运算符将 c+b 的值与字符串“
”连接，在输出 c+b 的值后执行换行操作。

2.7.3　赋值运算符

赋值运算符主要用于处理表达式的赋值操作，PHP 中提供了很多赋值运算符，其用法及意



义如表 2.7 所示。

表 2.7　常用的赋值运算符

操　作	符　号	实　例	展开形式	意　义
赋值	=	$a=b	$a=3	将右边的值赋给左边
加	+=	$a+= b	$a=$a + b	将右边的值加到左边
减	-=	$a-= b	$a=$a-b	将右边的值减到左边
乘	*=	$a*= b	$a=$a * b	将左边的值乘以右边
除	/=	$a/= b	$a=$a / b	将左边的值除以右边
连接字符	.=	$a.= b	$a=$a. b	将右边的字符加到左边
取余数	%	$a%= b	$a=$a % b	将左边的值对右边取余数

下面举一个非常简单的赋值运算符的例子，即为变量赋值：

```
$a=5;
```

此处应用“=”运算符为变量 a 赋值，下面再举一个稍微复杂的例子，代码如下：

```
<?php
$a=5;            //使用“=”运算符为变量a赋值
$b=10;           //使用“=”运算符为变量b赋值
$a*=$b;          //使用“*=”运算符获得变量a乘以变量b的值，并赋给变量a
echo $a;         //输出重新赋值后变量a的值
?>
```

运行结果为：50

多学两招：

在执行 i=i+1 的操作时，建议使用 i+=1 来代替。因为其符合 C/C++的习惯，且执行效率较高。

2.7.4　位运算符

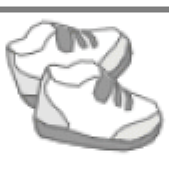
位运算符是指对二进制位从低位到高位对齐后进行运算。在 PHP 中的位运算符如表 2.8 所示。

表 2.8　位运算符

符　号	作　用	举　例
&	按位与	$m & $n
|	按位或	$m | $n
^	按位异或	$m ^ $n
~	按位取反	$m ~ $n
<<	向左移位	$m << $n
>>	向右移位	$m >> $n

例 2.13　使用位运算符对变量中的值进行位运算操作，实例代码如下：（**实例位置：配套资源\mr\2\example\2.13**）

```
<?php
$m = 8 ;                        //运算时会将8转换为二进制码1000
$n = 12 ;                       //运算时会将12转换为二进制码1100
$mn = $m&$n ;                   //将1000和1100做与操作后转换为十进制码
```



Note

```
echo $mn ."<br>";                //输出转换结果
$mn = $m | $n ;                  //将1000和1100做或操作后转换为十进制码
echo $mn ."<br>";                //输出转换结果
$mn = $m ^ $n ;                  //将1000和1100做异或操作后转换为十进制码
echo $mn ."<br>";                //输出转换结果
$mn = ~$m ;                      //将1000做非操作后转换为十进制码
echo $mn ."<br>";                //输出转换结果
?>
```

运行结果为：8　12　4　-9

2.7.5　自增或自减运算符

自增运算符“++”和自减运算符“--”与算术运算符相似，都是对数值型数据进行操作，但算术运算符适合在有两个或两个以上不同操作数的场合使用，当只有一个操作数时，就可以使用“++”或“--”运算符。

例 2.14 举个简单的例子，来加深对自增和自减运算符的理解，具体代码如下：**（实例位置：配套资源\mr\2\example\2.14）**

```
<?php
$a=10;
$b=5;
$c=8;
$d=12;
echo "a=".$a."  b=".$b."  c=".$c."  d=".$d."<br>";   //输出上面四个变量的值， 是空格符
echo "++a=".++$a."<br>";         //计算变量a自加的值
echo "b++=".$b++."<br>";         //计算变量b自加的值
echo "--c=".--$c."<br>";         //计算变量c自减的值
echo "d--=".$d--."<br>";         //计算变量d自减的值
?>
```

运行结果如图 2.23 所示。

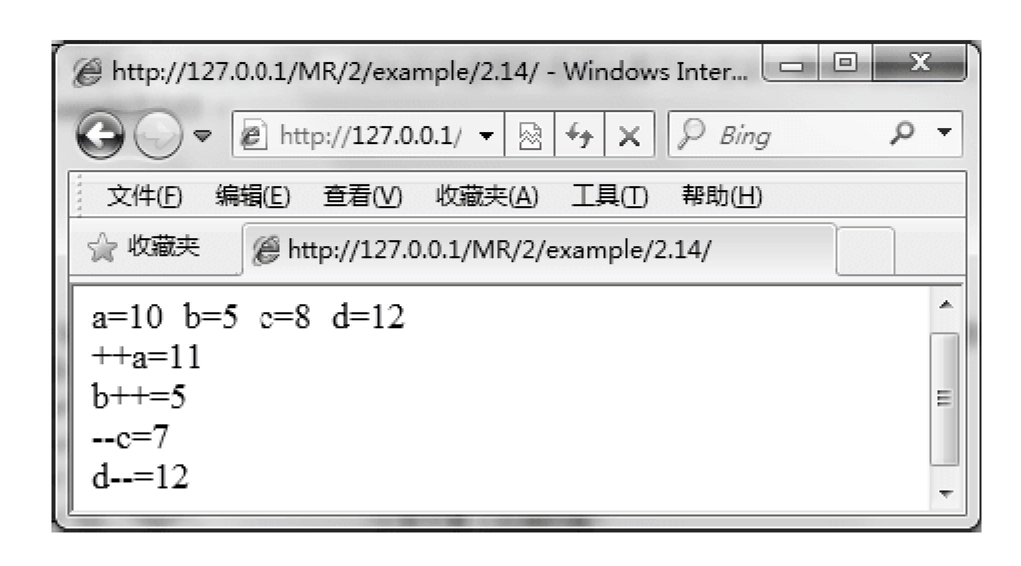
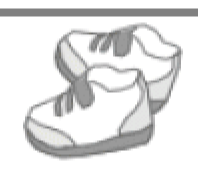
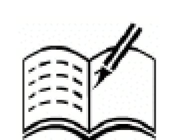


图 2.23　自增和自减运算符示例运行结果

指点迷津：

例 2.14 中变量$b 自加和$d 自减后的值为什么未发生改变？原因在于：当运算符位于变量前时（++$a），先自加，然后再返回变量的值；当运算符位于变量后时（$a++），先返回变量的值，然后再自加，即输出的是变量 a 的值，并非 a++的值。





2.7.6 逻辑运算符

逻辑运算符用于处理逻辑运算操作，是程序设计中一组非常重要的运算符。PHP 的逻辑运算符如表 2.9 所示。

表 2.9 PHP 的逻辑运算符

运 算 符	实 例	结 果
&&或 and（逻辑与）	$m and $n 或$m && $n	当$m 和$n 都为真或假时，返回 True 或 False 当$m 和$n 有一个为假时，返回 Flase
||或 or（逻辑或）	$m || $n 或$m or $n	当$m 和$n 都为真或假时，返回 True 或 False 当$m 和$n 有一个为真时，返回 True
Xor（逻辑异或）	$m xor $n	当$m 和$n 都为真或假时，返回 True 或 False 当$m 和$n 有一个为真时，返回 True
!（逻辑非）	!$m	当$m 为假时返回 True，当$m 为真时返回 False

例 2.15 使用逻辑运算符判断如果变量存在，且值不为空，则执行数据的输出操作，否则弹出提示信息（变量值不能为空!）。具体代码如下：（**实例位置：配套资源\mr\2\example\2.15**）

```
<?php
$a="";                              //如果变量a的值为空则输出提示信息,否则输出"明日科技欢迎您!"
if(isset($a) && !empty($a)){  //使用and判断变量a和变量b
    echo "明日科技欢迎您！ ";
}else{
    echo "<script>alert('变量值不能为空！ ');</script>";
}
?>
```

运行结果如图 2.24 所示。



图 2.24 使用逻辑与判断变量的真假

指点迷津：

例 2.15 在 if 语句中，应用逻辑与判断当变量存在，且值不为空的情况下输出数据，否则输出提示信息。

isset()函数检查变量是否设置，如果设置则返回 True，否则返回 False。

empty()函数检测变量是否为空，如果为空则返回 True，否则返回 False。

2.7.7 比较运算符

比较运算符主要用于比较两个数据的值，返回值为一个布尔类型。PHP 中的比较运算符如



表 2.10 所示。

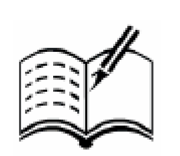


表 2.10　PHP 的比较运算符

运　算　符	实　　例	结　　果
<（小于）	$m<$n	当$m 小于$n 时，返回 True，否则返回 False
>（大于）	$m>$n	当$m 大于$n 时，返回 True，否则返回 False
<=（小于等于）	$m<=$n	当$m 小于等于$n 时，返回 True，否则返回 False
>=（大于等于）	$m>=$n	当$m 大于等于$n 时，返回 True，否则返回 False
= =（相等）	$m= =$n	当$m 等于$n 时，返回 True，否则返回 False
!=（不等）	$m!=$n	当$m 不等于$n 时，返回 True，否则返回 False
= = =（恒等）	$m=== $n	当$m 等于$n，并且数据类型相同，返回 True，否则返回 False
!==（非恒等）	$m!==$n	当$m 不等于$n，并且数据类型不相同，返回 True，否则返回 False

表 2.10 中的===和!==较少见。

例 2.16　使用比较运算符比较小刘与小李的工资，具体代码如下：**（实例位置：配套资源\mr\2\example\2.16）**

```
<?php
$a=2150;                                    //小刘的工资2150
$b=2240;                                    //小李的工资2240
echo "a=".$a."  b=".$b."<br>";
echo "a < b的返回值为： ";
echo var_dump($a<$b)."<br>";                //比较a是否小于b
echo "a >= b的返回值为： ";
echo var_dump($a>=$b)."<br>";               //比较a是否大于等于b
echo "a == b的返回值为： ";
echo var_dump($a==$b)."<br>";               //比较a是否等于b
echo "a != b的返回值为： ";
echo var_dump($a!=$b)."<br>";               //比较a是否不等于b
?>
```

运行结果如图 2.25 所示。

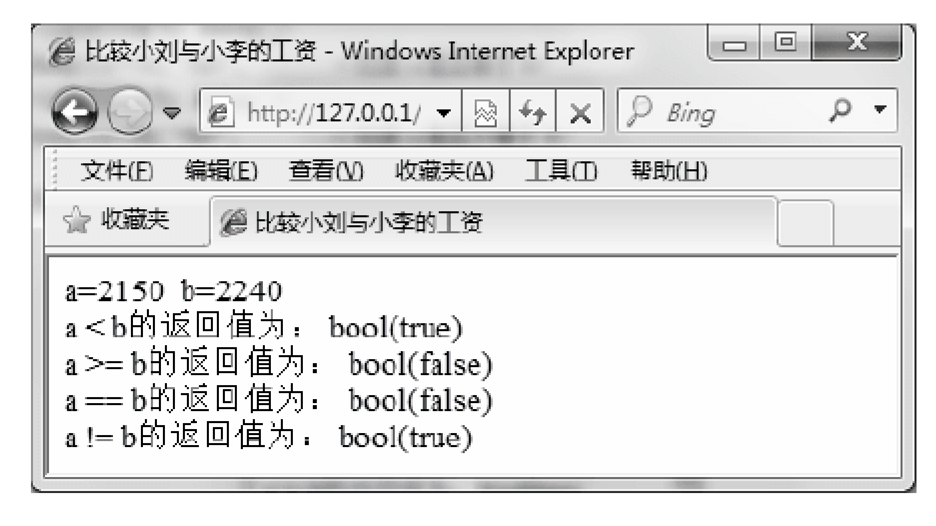
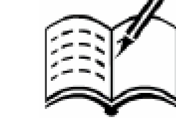


图 2.25　比较运算符示例运行结果

指点迷津：

例 2.16 中使用 var_dump()函数来输出变量对比后返回的值，其目的是让读者更直观地看到返回结果。





2.7.8　三元运算符

三元运算符可以提供简单的逻辑判断，其应用格式为：

```
表达式1?表达式2:表达式3
```

如果表达式 1 的值为 True，则执行表达式 2，否则执行表达式 3。

例 2.17　通过三元运算符定义分页变量的值，具体代码如下：（**实例位置：配套资源\mr\2\example\2.17**）

```
<?php
//通过三元运算符判断分页变量page的值，如果变量存在，则直接输出变量值；否则，为变量赋值1
$page=(isset($_GET['page']))?$_GET['page']:"1";
echo $page;             //输出变量值
?>
<a href="index.php?page=2">分页超级链接</a>
```

运行结果如图 2.26 所示。

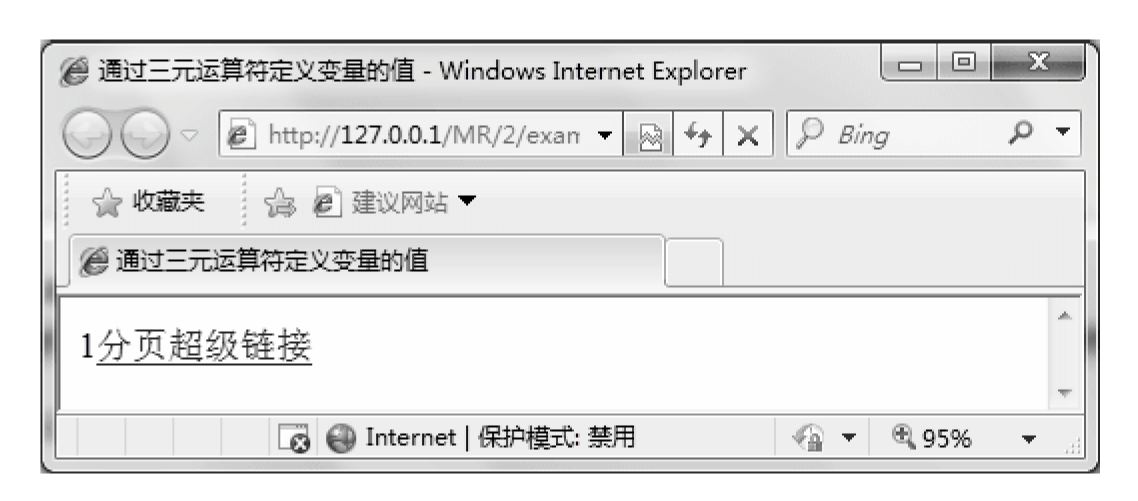


图 2.26　三元运算符

指点迷津：

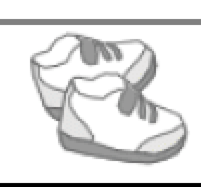
例 2.17 中介绍的方法在项目的实际开发中非常实用，特别是在分页技术中，根据超链接传递的参数值定义分页变量。其原理是：首先应用 isset()函数检测$_GET['page']全局变量是否存在，如果存在则直接将该值赋给变量 page；否则为变量 page 赋值 1。

2.7.9　运算符的使用规则

所谓运算符的使用规则就是当表达式中包含多种运算符时运算符的执行顺序，这与数学四则运算中的先计算乘除后计算加减是相同的道理。PHP 的运算符优先级如表 2.11 所示。

表 2.11　运算符的优先级

优先级别	运　算　符	优先级别	运　算　符
1	or，and，xor	9	++，--
2	赋值运算符	10	+，--（正、负号运算符），!，~
3	||，&&	11	= =，!=，<>
4	|，^	12	<，<=，>，>=
5	&，.	13	?:
6	+，-	14	->
7	/，*，%	15	=>
8	<<，>>		





Note

多学两招：

对于表 2-11 中众多的优先级别，如果想全部记住是不太现实的，况且也没有这个必要。如果表达式确实很复杂，而且包含较多的运算符，不妨多加（ ），如$a and (($b != $c) or (5 * (50 - $d)))。这样就会减少出现逻辑错误的可能性。

上机演练

上机演练 7　比较两个时间的大小

在一些程序中经常需要将两个时间进行比较，但是由于时间是由年、月、日、时、分、秒组成的，比较起来很不方便，这时可以把时间转换成时间戳来进行比较。通过 date()、strtotime()和 ceil()函数实现比较两个时间戳的大小，运行结果如图 2.27 所示。

上机演练 8　判断数字的奇偶性

条件运算符是进行或真或假运算的。下面通过三元运算符判断数字的奇偶性，运行结果如图 2.28 所示。

1305012224 输出日期2011-05-10 15:23:44

1304524800 输出日期2011-05-05 00:00:00

距2011年5月5号已过去6天

图 2.27　比较两个时间戳的大小

0是偶数 1是奇数 2是偶数 3是奇数
5是奇数 6是偶数 7是奇数 8是偶数 9是奇数

图 2.28　用三元运算符判断数字的奇偶性

上机演练 9　通过逻辑运算符判断用户的权限

逻辑运算符往往作为 if 语句的条件出现。下面通过逻辑运算符和 if 语句来判断用户是否具有后台管理权限，运行结果如图 2.29 和图 2.30 所示。



图 2.29　具有后台管理权限

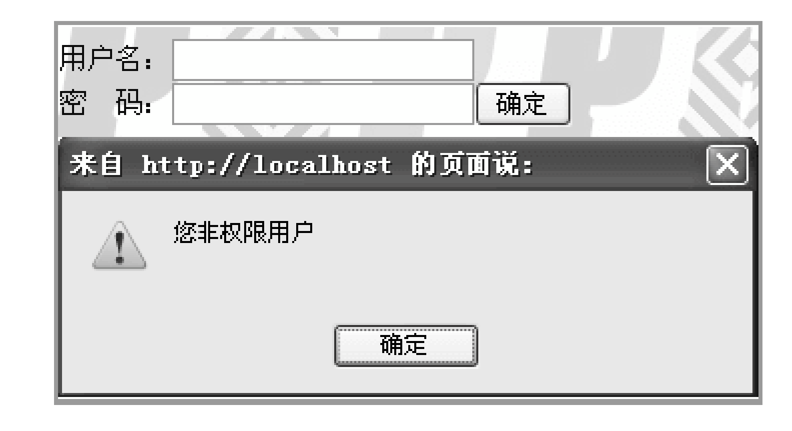


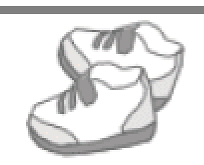
图 2.30　不具有后台管理权限

上机演练 10　获取 2000～2020 年之间的所有闰年

在 PHP 语言中，为了提升编程效率设置了一些自增自减运算符，这种运算符在循环语句中得到广泛应用。应用自增自减运算符、算术运算符和 for 循环语句获取 2000～2020 年之间的所有闰年，运行结果如图 2.31 所示。



图 2.31　打印 2000～2020 年之间的所有闰年



本章摘要



1．PHP 的原理。
2．PHP 的标记。
3．PHP 注释的使用。
4．常量的声明、使用以及预定义常量的用法。
5．变量的声明，如何命名、如何赋值以及变量的作用域。
6．PHP 中的数据类型、类型之间的转换以及类型的检测。
7．PHP 中各种运算符的使用方法及规则。

习 题

1．声明变量时，变量的名称以（　　）符号开头。
A．&　　B．*　　C．$　　D．￥

2．PHP 中的标量数据类型有（　　）。
A．字符串型（string）　　B．整型（int）
C．浮点型（float）　　D．布尔型（boolean）

3．不等于运算符是（　　）。
A．!=　　B．==　　C．===　　D．!==

4．取余数运算符的符号是（　　）。
A．&　　B．%　　C．￥　　D．#

5．检测数据类型是否为字符串型的函数是（　　）。
A．is_bool　　B．is_string　　C．is_float　　D．is_boolean

6．使用传值赋值方式，为$b 赋值，请将下列代码补充完整。

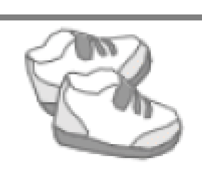
```
<?php
$a=10;
        (    )
echo $a;
echo $b;
?>
```

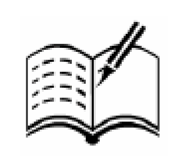
7．连接字符串的运算符是（　　）。

8．以下程序的输出结果是（　　）。

```
<?php
$b=201;
       $c=40;
       $a=$b>$c?4:5;
       echo $a;
?>
```

9．执行程序段<?php echo 8%(-2) ?>将输出（　　）。





Note

10．定义常量的函数是（　　）。

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 1 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　自定义数字加密算法

运算符是表达式的组成部分，没有运算符的表达式是不存在的。通过算术运算符设置数字的加密算法，运行结果如图 2.32 和图 2.33 所示。



图 2.32　口令加密



图 2.33　口令解密

实战模拟 2　随机输出字符串

在应用字符串时，往往需要将两个字符串进行连接。这时可以使用字符串运算符。通过字符串运算符打印随机组合生日祝福语，运行结果如图 2.34 所示。

生日快乐祝你万事如意

图 2.34　打印随机组合生日祝福语

第 3 章

PHP 函数

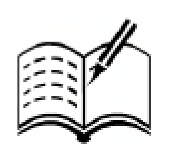
（自学视频、源程序：配套资源\mr\3\）

函数分为系统内部函数和用户自定义函数两种。在日常开发中，如果有一个功能或一段代码要经常使用，就可以把它写成自定义函数，在需要时随时进行调用。除了自定义函数外，PHP 还提供了庞大的函数库，包含有几千种的内置函数，可以直接使用它们实现相应的功能。在程序中调用函数的目的是为了简化编程，减少代码量，提高效率，达到增加代码重用性、避免重复开发的目的。

学习摘要：

- PHP 函数的定义、调用以及参数的传递
- PHP 变量函数库中的经典函数
- PHP 字符串函数库中的经典函数
- PHP 日期时间函数库中的经典函数
- PHP 数学函数库中的经典函数
- PHP 文件系统函数库中的经典函数
- MySQL 函数库中的经典函数

3.1　PHP 函数



视频讲解：配套资源\mr\3\video\PHP 函数.exe

在开发过程中，经常要重复某种操作或处理，如数据查询、字符操作等，如果每个模块的操作都要重新输入一次代码，不仅令程序员头痛不已，而且对于代码的后期维护及运行效果也有着较大的影响，使用 PHP 函数即可让这些问题迎刃而解。

3.1.1　定义和调用函数

函数，就是将一些重复使用的功能写在一个独立的代码块中，在需要时单独调用。创建函数的基本语法格式为：

```
function fun_name($str1,$stgr2…$strn){
    fun_body;
}
```

参数说明：function：为声明自定义函数时必须使用到的关键字。fun_name：为自定义函数的名称。$str1…$strn：为函数的参数。fun_body：为自定义函数的主体，是功能实现部分。

当函数被定义后，所要做的就是调用这个函数。调用函数的操作十分简单，只需要引用函数名并赋予正确的参数即可完成函数的调用。

例 3.1　定义函数 example()，计算传入的参数的平方，然后连同表达式和结果全部输出，代码如下：（**实例位置：配套资源 mr\3\example\3.1**）

```
<?php
    /*    声明自定义函数    */
    function example($num){
        return "$num * $num = ".$num * $num;            //返回计算后的结果
    }
    echo example(10);                                   //调用函数
?>
```

运行结果为：10 * 10 = 100

3.1.2　在函数间传递参数

在调用函数时需要向函数传递参数，被传入的参数称为实参，而函数定义的参数称为形参。参数传递的方式有按值传递方式、按引用传递方式和默认参数 3 种。

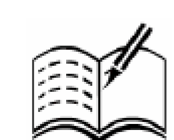
1. 按值传递方式

按值传递方式是指将实参的值复制到对应的形参中，在函数内部的操作针对形参进行，操作的结果不会影响到实参，即函数返回后，实参的值不会改变。

例 3.2　首先定义一个函数 example()，功能是将传入的参数值做一些运算后再输出。接着在函数外部定义一个变量$m，也就是要传进来的参数。最后调用函数 example($m)，输出函数的返回值$m 和变量$m 的值，代码如下：（**实例位置：配套资源 mr\3\example\3.2**）

```
<?php
function example( $m ){                    //定义一个函数
```





```
        $m = $m * 5 + 10;
    echo "在函数内：\$m = ".$m;                //输出形参的值
    }
    $m = 1;
    example( $m ) ;                           //传递值，将$m的值传递给形参$m
    echo "<p>在函数外  \$m = $m <p>" ;         //实参的值没有发生变化，输出m=1
    ?>
```

运行结果如图 3.1 所示。

2. 按引用传递方式

按引用传递方式就是将实参的内存地址传递到形参中。这时，在函数内部的所有操作都会影响到实参的值，返回后实参的值会发生变化。引用传递方式就是传值时在原基础上加“&”符即可。

例 3.3 仍然使用例 3.2 中的代码，唯一不同的地方就是多了一个“&”符，代码如下：（**实例位置：配套资源 mr\3\example\3.3**）

```
    <?php
    function example( &$m ){                  //定义一个函数，同时传递参数$m的变量
     $m = $m * 5 + 10;
    echo "在函数内：\$m = ".$m;                //输出形参的值
    }
    $m = 1;
    example( $m ) ;                           //传递值：将$m的值传递给形参$m
    echo "<p>在函数外：\$m = $m <p>" ;         //实参的值发生变化，输出m=15
    ?>
```

运行结果如图 3.2 所示。

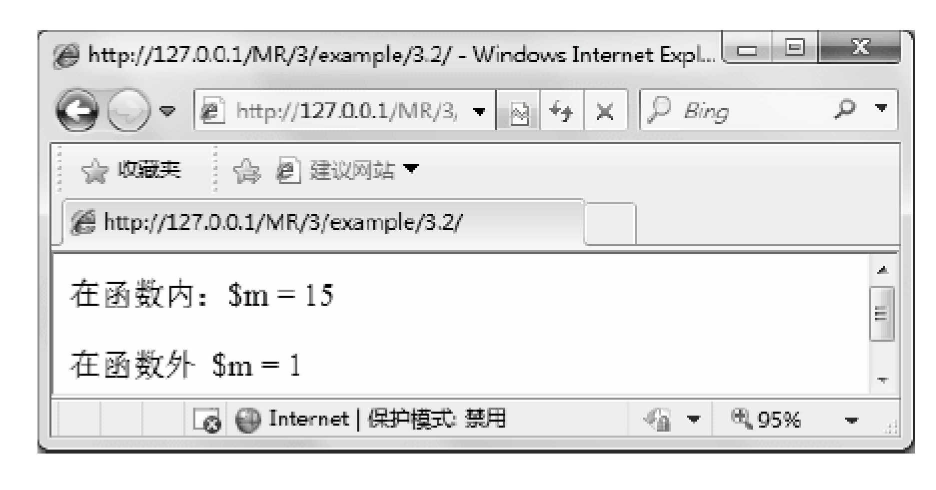


图 3.1 按值传递

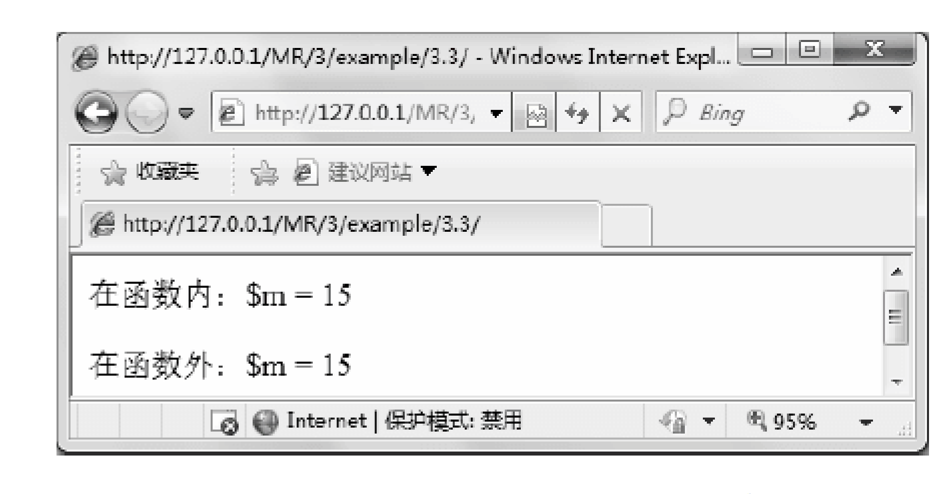


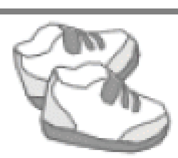
图 3.2 按引用传递方式

3. 默认参数（可选参数）

还有一种设置参数的方式，默认参数即可选参数。可以指定某个参数为可选参数，将可选参数放在参数列表末尾，并且指定其默认值为空。

例 3.4 使用可选参数实现一个简单的价格计算功能，设置自定义函数 values 的参数$tax 为可选参数，其默认值为空。第一次调用该函数，并且为参数$tax 赋值 0.25，输出价格；第二次调用该函数，不为参数$tax 赋值，输出价格，代码如下：（**实例位置：配套资源 mr\3\example\3.4**）

```
    <?php
        function values($price,$tax=""){          //定义一个函数，其中的一个参数初始值为空
            $price=$price+($price*$tax);          //声明一个变量$price，等于两个参数的运算结果
            echo "价格:$price<br>";               //输出价格
        }
```



```
        values(100,0.25);                //为可选参数赋值0.25
        values(100);                     //没有为可选参数赋值
    ?>
```

运行结果为：价格:125　　价格:100



脚下留神：

当使用默认参数时，默认参数必须放在非默认参数的右侧，否则函数可能出错。

指点迷津：

从 PHP5 开始，默认值也可以通过引用传递。

3.1.3　从函数中返回值

前面介绍了如何定义和调用一个函数，并且讲解了如何在函数间传递值，这里将讲解函数的返回值。通常，函数将返回值传递给调用者的方式是使用关键字 return。

return 将函数的值返回给函数的调用者，即将程序控制权返回到调用者的作用域。如果在全局作用域内使用 return 关键字，那么将终止脚本的执行。

例 3.5　使用 return()函数返回一个操作数。先定义函数 values，函数的作用是输入物品的单价、重量，然后计算总金额，最后输出商品的价格，代码如下：**（实例位置：配套资源 mr\3\example\3.5）**

```
    <?php
        function values($price,$tax=0.45){      //定义一个函数，函数中的一个参数有默认值
            $price=$price+($price*$tax);        //计算物品金额
            return $price;                      //返回金额
        }
        echo values(100);                       //调用函数
    ?>
```

运行结果为：145

指点迷津：

return 语句只能返回一个参数，也即只能返回一个值，不能一次返回多个。如果要返回多个结果，就要在函数中定义一个数组，将返回值存储在数组中返回。

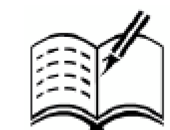
3.1.4　变量函数

PHP 支持变量函数。那么什么是变量函数？看下面的示例。

例 3.6　首先定义 3 个函数，接着声明一个变量，通过变量来访问不同的函数，代码如下：**（实例位置：配套资源 mr\3\example\3.6）**

```
    <?php
    function come() {                       //定义come()函数
        echo "来了<p>";
    }
    function go($name = "jack") {           //定义go()函数
        echo $name.'走了<p>';
```





```
    }
    function back($string){                //定义back()函数
        echo "又回来了，$string<p>";
    }
    $func = 'come';                        //声明一个变量，将变量赋值为“come”
    $func();                               //使用变量函数来调用函数come()
    $func = 'go';                          //重新为变量赋值
    $func('Tom');                          //使用变量函数来调用函数go()
    $func = 'back';                        //重新为变量赋值
    $func('Lily');                         //使用变量函数来调用函数back()
    ?>
```

运行结果如图 3.3 所示。

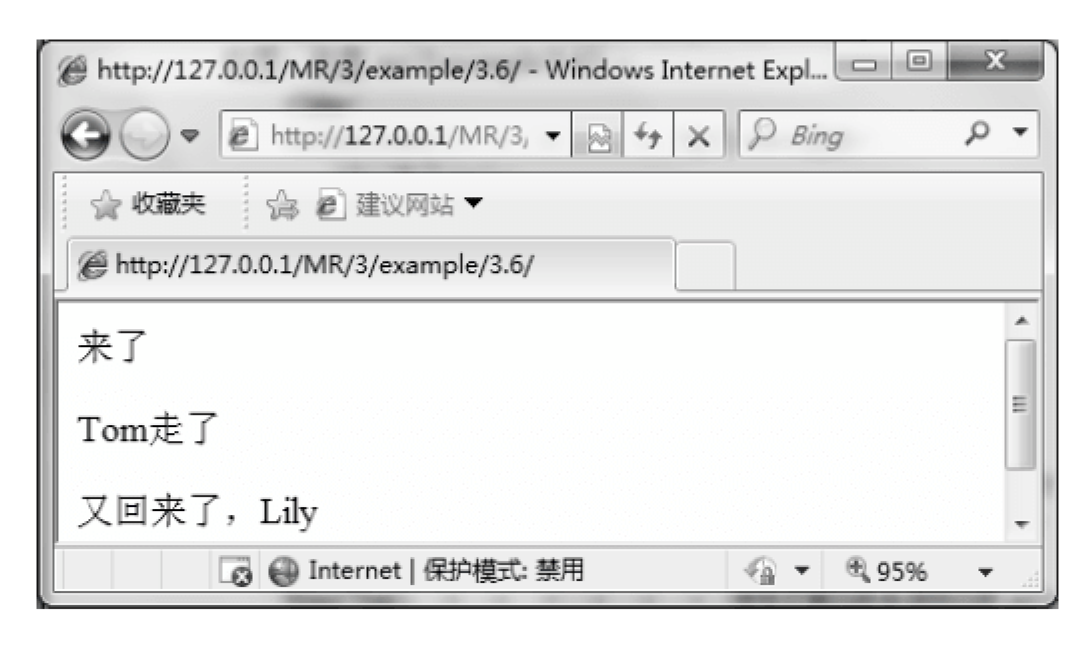


图 3.3 变量函数

可以看到，函数的调用是通过改变变量名来实现的，通过在变量名后面加上小括号，PHP 将自动寻找与变量名相同的函数，并且执行它。如果找不到对应的函数，系统将会报错。这就是变量函数。

3.1.5 对函数的引用

按引用传递参数可以修改实参的内容。引用不仅可用于普通变量、函数参数，也可作用于函数本身。对函数的引用，其实就是对函数返回结果的引用。

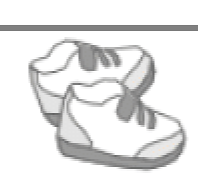
例 3.7 首先定义一个函数，在函数名前加“&”符。接着通过变量$str 引用该函数，最后输出变量$str，实际上就是$tmp 的值，代码如下：（**实例位置：配套资源 mr\3\example\3.7**）

```
    <?php
    function &example($tmp=0){              //定义一个函数，注意加“&”符
        return $tmp;                        //返回参数$tmp
    }
    $str = &example("看到了");              //声明一个函数的引用$str
    echo $str."<p>";                        //输出$str
    ?>
```

运行结果为：看到了

脚下留神：

和参数传递不同，这里必须在两个地方使用“&”符，用来说明返回的是一个引用。





Note

3.1.6 取消引用

当不再需要引用时，可以取消引用。取消引用使用 unset()函数，它只是断开了变量名和变量内容之间的绑定，而不是销毁变量内容。

例 3.8 首先声明一个变量和变量的引用，输出引用后取消引用，再次调用引用和原变量。可以看到，取消引用后对原变量没有任何影响，代码如下：**（实例位置：配套资源 mr\3\example\3.8）**

```
<?php
$num = 1234;                          //声明一个整型变量
$math = &$num;                        //声明一个对变量$num的引用$math
echo "\$math is:   ".$math."<br>";    //输出引用$math
unset($math);                         //取消引用$math
echo "\$num is:   ".$num;             //输出原变量
?>
```

运行结果为：$math is: 1234　　　$num is: 1234

上机演练

上机演练 1　自定义函数过滤字符串

用户在论坛上发表或回复帖子时可能会遇到这种情况：当单击“发布”按钮时，系统提示使用了禁用词语，禁止发布。这是论坛的开发人员为了规范论坛而采用的过滤字符串的手段，即通过自定义函数实现过滤字符串。运行结果如图 3.4 所示。

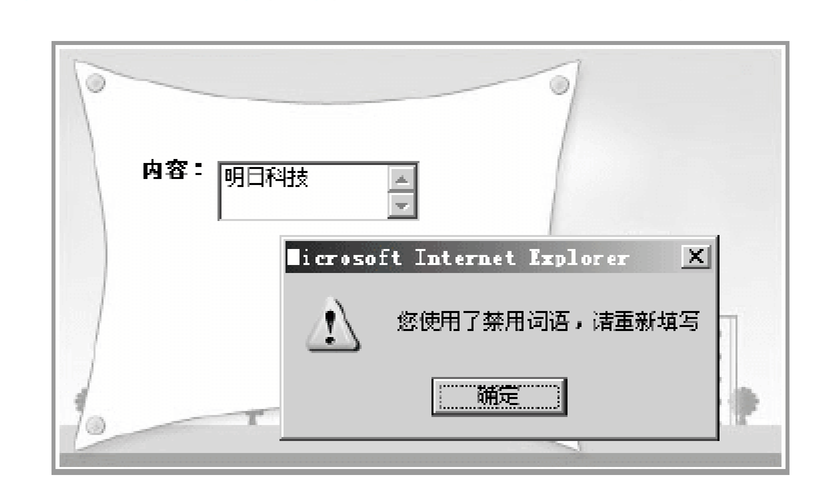


图 3.4　自定义函数过滤字符串

3.2　PHP 变量函数库

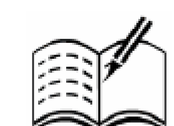
视频讲解：配套资源\mr\3\video\PHP 变量函数库.exe

除了用户自行编写的函数库外，PHP 自身也提供了很多内置的函数，PHP 变量函数库就是其中一个。但并不是所有的函数都会经常用到，因此，读者只要熟悉一些常用的即可。这里将根据实际开发的需求，向读者介绍一些常用的变量函数，如表 3.1 所示。

表 3.1　常用的变量函数介绍

类　型	说　明
empty()	检查一个变量是否为空，为空，返回 True；否则返回 False
gettype()	获取变量的类型
intval()	获取变量的整数值





续表

类　　型	说　　明
is_array()	检查变量是否为数组类型
is_int()	检查变量是否为整数
is_numeric()	检查变量是否为数字或由数字组成的字符串
isset()	检查变量是否被设置，即是否被赋值
print_r()	打印变量
settype()	设置变量的类型，可将变量设为另一个类型
unset()	释放给定的变量，即销毁这个变量
var_dump()	打印变量的相关信息

isset()函数检查变量是否被设置，设置则返回 True，否则返回 False。其语法如下：

```
bool isset ( mixed var [, mixed var [, ...]])
```

参数 var 必选参数，输入的变量；参数 var2 可选参数，此参数是输入的变量，可有多个。

例 3.9　应用 isset()函数和 empty()函数判断用户提交的用户名和密码是否为空，代码如下：**（实例位置：配套资源\mr\3\example\3.9）**

```
<?php
if(isset($_POST['Submit']) && $_POST['Submit']=="登录"){//通过isset()函数对登录按钮进行判断
$user=$_POST['user'];                           //通过$_POST函数调用表单文本域的值
$pass=$_POST['pass'];
    if(empty($user)||empty($pass)){             //通过if语句判断用户名或密码不能为空
        echo "<script>alert('用户名或密码不能为空');</script>";//当用户名或密码为空时，给出提示
    }
}
?>
```

运行结果如图 3.5 所示。

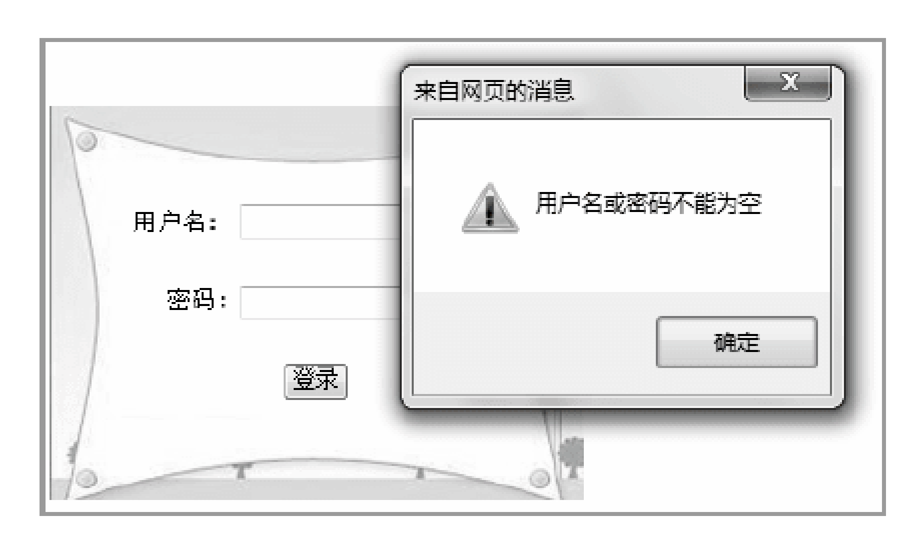


图 3.5　用户名或密码不能为空

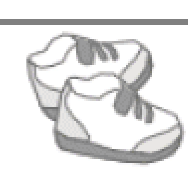
脚下留神：

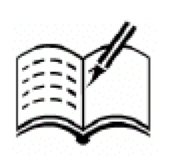
isset()函数只能用于变量，因为传递任何其他参数都将造成解析错误。若想检测常量是否已设置，可使用 defined()函数。

3.3　PHP 字符串函数库

视频讲解：配套资源\mr\3\video\PHP 字符串函数库.exe

PHP 字符串函数库是 PHP 开发中一项非常重要的内容，用户必须掌握其中常用函数的使





用方法。在表 3.2 中对 PHP 常用的字符串函数进行了总结。

表 3.2 常用字符串函数

函　数	功　能
addcslashes()	实现转义字符串中的字符，即在指定的字符前面加上反斜线
explode()	将字符串依指定的字符串或字符 separator 切开
echo()	用来输出字符串
ltrim()	删除字符串开头的连续空白
md5()	计算字符串的 md5 哈希
strlen()	获取指定字符串的长度
str_ireplace()	将某个指定的字符串都替换为另一个指定的字符串（大小写不敏感）
str_repeat()	将指定的字符串重复输出
str_replace()	取代所有在字符串中出现的字串
strchr()	获取指定字符串（A）在另一个字符串（B）中首次出现的位置
stristr()	获取指定字符串（A）在另一个字符串（B）中首次出现的位置到（B）字符串末尾的所有字符串
strstr()	获取一个指定字符串在另一个字符串中首次出现的位置到后者末尾的子字符串
substr_replace()	将字符串中的部分字符串替换为指定的字符串
substr()	从指定的字符串 str 中按照指定的位置 start 截取一定长度 length 的字符

1. explode()函数

explode()函数将字符串依指定的字符串或字符 separator 切开。其语法如下：

```
array explode(string separator, string string, [int limit])
```

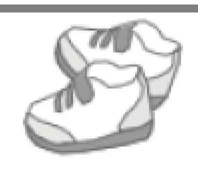
返回由字符串组成的数组，每个元素都是 string 的一个子串，它们被字符串 separator 作为边界点分隔出来。

如果设置了 limit 参数，则返回的数组包含最多 limit 个元素，而最后那个元素将包含 string 的剩余部分；如果 separator 为空字符串（""），explode()函数将返回 False；如果 separator 所包含的值在 string 中找不到，那么 explode()函数将返回包含 string 单个元素的数组；如果参数 limit 是负数，则返回除了最后的-limit 个元素外的所有元素。

例 3.10 在电子商务网站的购物车中，通过特殊标识符“@”将购买的多种商品组合成一个字符串存储在数据表中，在显示购物车中的商品时，以“@”作为分割的标识符进行拆分，将商品字符串分割成 N 个数组元素，最后通过 foreach 循环语句输出数组元素，即输出购买的商品，代码如下：**（实例位置：配套资源 mr\3\example\3.10）**

```
<?php
$str="品牌电脑@品牌手机@高档男士衬衫@高档女士挎包";        //定义字符串常量
$str_arr=explode("@",$str);                              //应用标识@分割字符串
foreach($str_arr as $key=>$value){                 //使用 foreach 语句遍历数组，输出键和值
  echo $value."<br>";                                    //输出商品
}
?>
```

运行结果如图 3.6 所示。



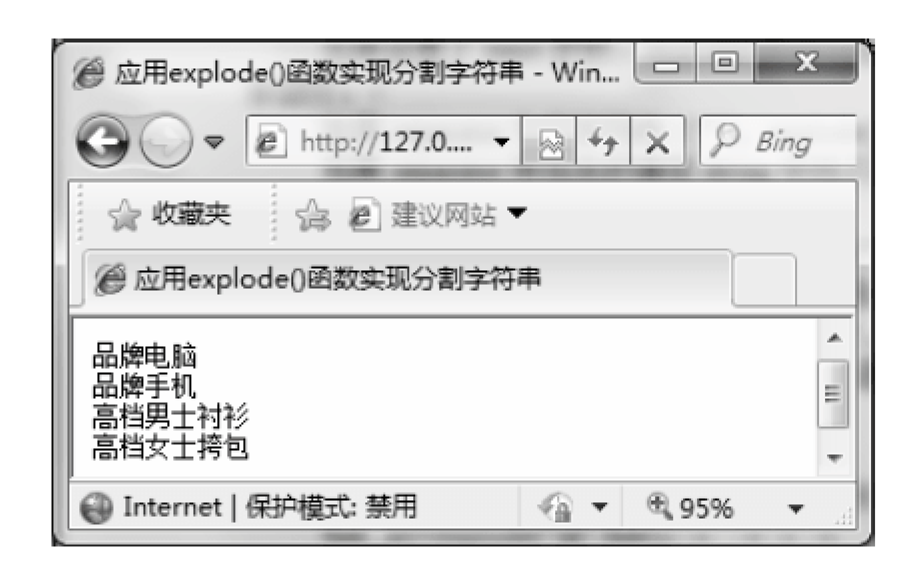
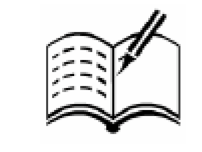


图 3.6　循环输出购物车中的商品



2. md5()函数

md5()函数计算字符串的 md5 哈希，该函数是一种编码的方式，但是不能解码。其语法如下：

```
string md5 ( string str , bool raw_output )
```

参数 str 为被加密的字符串；参数 raw_output 为布尔型，返回 True 表示加密字符串以二进制格式返回。

例如：应用 md5()函数对字符串“明日科技”进行编码，代码如下：

```
<?php
    echo md5("明日科技");    //63bac9345fcfdf1ec8cfa82f1c996c29
?>
```

运行结果为：63bac9345fcfdf1ec8cfa82f1c996c29

多学两招：

通常为了保证用户注册信息的安全，可用 md5()函数对用户注册的密码进行加密。

上机演练

上机演练 2　获取上传文件的后缀

文件上传几乎是所有网站都必需的模块，然而其对于服务器来说是具有很大风险的，应该对文件大小、文件类型进行限制。下面通过字符串函数 strrev()获取上传文件的后缀，运行结果如图 3.7 所示。

上机演练 3　统一上传文件名称的大小写

统一上传文件名称的大小写是十分必要的。例如上传图片文件 img.jpg 和 Img.jpg，如果文件名称未经大小写统一，则上传时可能就不会出现覆盖提示。下面通过字符串函数 strtoupper()、strtolower()实现统一上传文件名称的大小写，运行结果如图 3.8 所示。

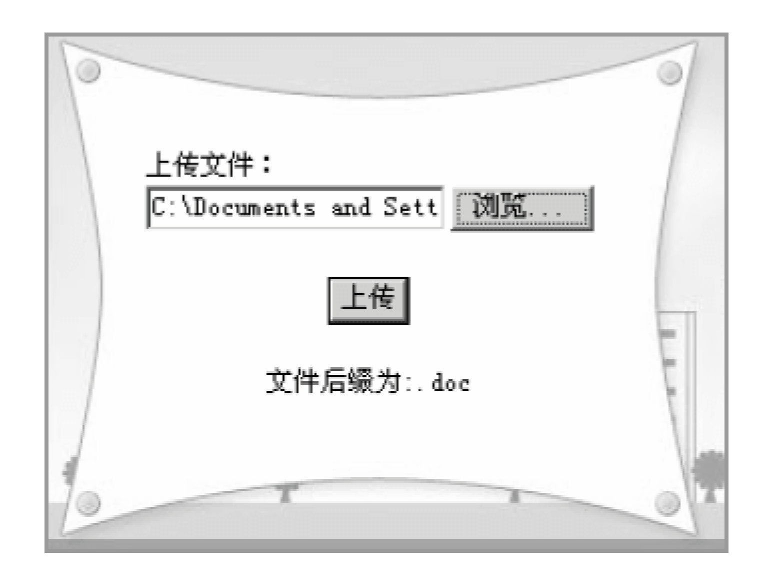
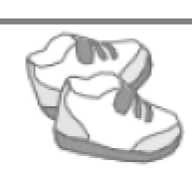


图 3.7　获取上传文件的后缀



图 3.8　统一上传文件名称的大小写



上机演练 4　论坛内容的简单输出

开发论坛程序中十分重要的一点是转换空格符和回车符，因为不是所有的论坛用户都知晓 HTML 标签“ 和
”的作用，大多数用户都喜欢用空格键和 Enter 键对文本进行控制，这就需要程序员人为地做到统一。下面通过自定义函数定义转换方法实现论坛内容的简单输出，运行结果如图 3.9 所示。

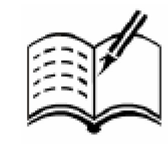




图 3.9　论坛内容的简单输出

3.4　PHP 日期时间函数库

视频讲解：配套资源\mr\3\video\PHP 日期时间函数库.exe

PHP 通过内置的日期和时间函数，完成对日期和时间的各种操作，其常用日期和时间函数如表 3.3 所示。

表 3.3　常用日期和时间函数介绍

函　　数	功　　能
checkdate()	验证日期的有效性
date()	格式化一个本地时间/日期
microtime()	返回当前 UNIX 时间戳和微秒数
mktime()	获取一个日期的 UNIX 时间戳
strftime()	根据区域设置格式化本地时间/日期
strtotime()	将任何英文文本的日期时间描述解析为 UNIX 时间戳
time()	返回当前的 UNIX 时间戳

1. checkdate()函数

checkdate()函数用于验证日期的有效性，如果日期有效则返回 True，否则返回 False。其语法如下：

```
bool checkdate ( int month, int day, int year)
```

参数 month 的有效值是从 1 到 12；参数 day 的有效值在给定的 month 所应该具有的天数范围之内，包括闰年；参数 year 的有效值是从 1 到 32767。

例如：应用 checkdate()函数判断日期是否有效，如果正确则输出 1，否则不输出。其代码如下：

```
<?php
$checkdate=checkdate(7,32,2008);      //判断日期是否有效
```





```
    echo $checkdate;                    //输出变量
    ?>
```

运行结果为：1

2. mktime()函数

mktime()函数用于返回一个日期的 UNIX 时间戳。其语法如下：

```
int mktime ( int hour, int minute, int second ,int month, int day ,int year , int is_dst)
```

mktime()函数根据给出的参数返回 UNIX 时间戳。时间戳是一个长整数，包含了从 UNIX 新纪元（1970 年 1 月 1 日）到给定时间的秒数。其参数可以从右向左省略，任何省略的参数会被设置成本地日期和时间的当前值。其中参数 is_dst 在夏令时可以被设为 1，如果不是则设为 0；如果不知道是否为夏令时则设为-1（默认值）。

例 3.11 应用 mktime()函数获取系统当前的时间戳，然后通过 date()函数来对其进行格式化，输出时间，代码如下：**（实例位置：配套资源 mr\3\example\3.11）**

```
<?php
echo "mktime函数返回的时间戳:". mktime()."<br>";
echo date( "M-d-Y", mktime());
?>
```

运行结果如图 3.10 所示。

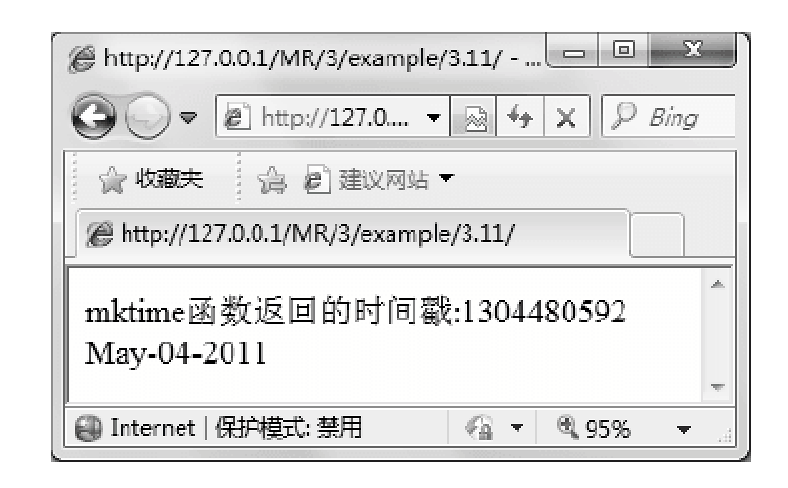


图 3.10 获取当前时间戳

上机演练

上机演练 5 日期、时间的格式化输出

在 PHP 程序设计中对日期、时间的格式化输出是经常用到的。下面通过函数 date()实现日期、时间的格式化输出，运行结果如图 3.11 所示。

上机演练 6 倒计时

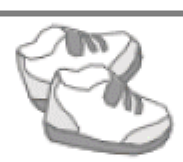
倒计时功能的实现同样也是将时间戳做算术运算。下面应用 strtotime()函数实现倒计时程序，运行结果如图 3.12 所示。



图 3.11 日期、时间的格式化输出



图 3.12 倒计时






Note

3.5 PHP 数学函数库

视频讲解：配套资源\mr\3\video\PHP 数学函数库.exe

PHP 提供了大量的内置数学函数，大大提高了开发人员在数学运算上的精准度。在表 3.4 中介绍了一些常用的 PHP 数学函数。

表 3.4 常用数学函数介绍

类 型	说 明
ceil()	返回不小于参数 value 值的最小整数，如果有小数部分则进一位
mt_rand()	返回随机数中的一个值
mt_srand()	配置随机数的种子
rand()	产生一个随机数，返回随机数的值
round()	实现对浮点数进行四舍五入
floor()	实现舍去法取整，该函数返回不大于参数 value 值的下一个整数，将 value 值的小数部分舍去取整
fmod()	返回除法的浮点数余数
getrandmax()	获取随机数最大的可能值
max()	返回参数中的最大值
min()	返回参数中的最小值

1. floor()函数

floor()函数实现舍去法取整，返回不大于参数 value 值的下一个整数，将 value 值的小数部分舍去取整。floor()函数返回的类型是浮点型，因为浮点型值的范围通常比整型要大。其语法如下：

```
float floor ( float value)
```

例如：应用 floor()函数对数值 6、7.2 和 8.9 进行取整。其代码如下：

```
<?php
echo floor(6). "<br>";
echo floor(7.2). "<br>";
echo floor(8.9). "<br>";
?>
```

运行结果为：6 7 8

2. fmod()函数

fmod()函数返回除法的浮点数余数。其语法如下：

```
float fmod ( float x, float y)
```

该函数返回被除数（x）除以除数（y）所得的浮点数余数。

例如：应用 fmod()函数获取 5 除以 1.5 所得的余数。其代码如下：

```
<?php
$x = 5;
$y = 1.5;
```



```
$z = fmod($x, $y);
echo $z;
?>
```

运行结果为：0.5

例 3.12 max()函数被广泛地应用于网站的最高访问量的统计或者销售系统中的销售统计。本例应用 max()函数获取一年中商品月销量最高的值。（**实例位置：配套资源 mr\3\example\3.12**）

（1）创建一个 form 表单，添加表单元素，用于提交一年中每个月的销售数据。

（2）在本页中编写 PHP 脚本，通过 isset()函数判断是否执行提交操作。如果执行提交操作，则通过$_POST[]全局数组获取每个月的销售数据，并赋给指定的变量，并且将这些数据存储到一个数组中，最终通过 max()函数统计数组中的最大值；如果未执行提交操作，那么为执行的变量赋空值。代码如下：

```
<?php
if(isset($_POST['Submit']) && $_POST['Submit']=="提交"){
    $month1=$_POST['month1'];
    $month2=$_POST['month2'];
//此处省略了部分代码
$array=array($_POST['month1'],$_POST['month2'],$_POST['month3'],$_POST['month4'],$_POST['month5'],$_POST['month6'],$_POST['month7'],$_POST['month8'],$_POST['month9'],$_POST['month10'],$_POST['month11'],$_POST['month12']);
    $max=max($array);
}else{
    $month1="";
    $month2="";
//此处省略了部分代码
    $max="";
}
?>
```

（3）输出月销售量最高的数据。运行结果如图 3.13 所示。

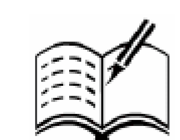


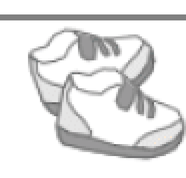
图 3.13　应用 max()函数获取月销量最高值

上机演练

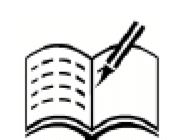
上机演练 7　中文图像验证码

验证码不但可以使用数字图像来完成，而且可以使用中文图像来完成。为了防止有些非法用户通过恶意程序登录站点，提高网站的安全性，这里通过中文图像来生成验证码，运行结果





如图 3.14 所示。



Note



图 3.14　中文图像验证码

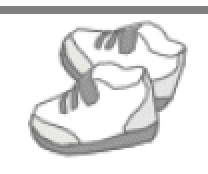
3.6　PHP 文件系统函数库

视频讲解：配套资源\mr\3\video\PHP 文件系统函数库.exe

对文件操作是存取数据的方式之一。相对于数据库来说，文件在使用上更方便、直接。如果数据较少、较简单，使用文件无疑是最合适的方法。PHP 对文件的操作是通过内置的文件操作系统函数来完成的。常用的文件系统函数如表 3.5 所示。

表 3.5　文件系统函数

函　　数	功　　能
basename()	返回文件路径中基本的文件名
copy()	将某文件由当前目录复制到其他目录，如果成功则返回 True，否则返回 False
file_exists()	判断指定的目录或文件是否存在。如果存在则返回 True，否则返回 False
file_put_contents()	将字符串写入到指定的文件中
file()	读取某文件的内容，并将结果保存到数组中，数组内每个元素的内容对应读取文件的一行
filetype()	返回文件的类型。可能的值有 fifo、char、dir、block、link、file 和 unknown
fopen()	打开某文件，并返回该文件的标识指针。该文件可以是本地的，也可以是远程的
fread()	从文件指针所指文件中读取指定长度的数据
is_dir()	如果该函数参数所代表的路径为目录并且该目录存在，则返回 True，否则返回 False
is_uploaded_file()	判断文件是否应用 HTTP POST 方式上传的，如果是则返回 True，否则返回 False
mkdir()	新建一个目录
move_uploaded_file()	应用 POST 方法上传文件
readfile()	读入一个文件，并将读入的内容写入到输出缓冲
rmdir()	删除指定的目录，如果删除成功则返回 True，否则返回 False
unlink()	用于删除文件，如果删除成功则返回 True，否则返回 False



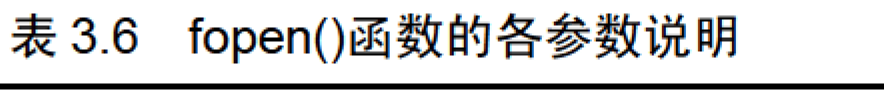


1. fopen()函数

fopen()函数用于打开某文件，并返回该文件的标识指针。该文件可以是本地的，也可以是远程的。其语法如下：

```
resource fopen(string filename, string mode [, int use_include_path [, resource context]])
```

fopen()函数中的各参数说明如表 3.6 所示。


表 3.6　fopen()函数的各参数说明

参　　数	说　　明
filename	必选参数。用于指定要打开文件的本地或远程地址
mode	必选参数。用于指定要打开文件的模式
use_include_path	可选参数。如果将该参数设置为 TRUE，则 PHP 会尝试按照 include_path 标准包含路径中的每个指向去打开文件
context	可选参数。设置提高文件流性能的一些选项

2. mkdir()函数

mkdir()函数用于判断某文件是否存在，并且是否可写。如果存在，并且可写则返回 True；否则返回 False。其语法如下：

```
bool mkdir(string pathname [, int mode])
```

参数 pathname 为必选参数，用于指定新建目录的名称；参数 mode 为可选参数，用于指定新建目录的模式。

例 3.13　就应用文件和目录的操作技术编写一个实例，实现文件、目录的创建和删除功能。**（实例位置：配套资源 mr\3\ example \3.13）**

创建 index.php 文件，首先判断文件夹和文件是否存在，如果存在则利用 file_get_contents()函数读取文本文件信息，然后删除文件夹及文件；如果不存在则使用 mkdir()和 fopen()函数创建文件夹和文件并向文件中写入文本信息。代码如下：

```
<?php
    if(!is_dir('txt')){                             //判断txt是否为文件夹目录
        mkdir('txt');                               //创建名为txt的文件夹目录
        $open=fopen("txt/in.txt","w+");             //以读写方式打开文件in.txt
            if(is_writable("txt/in.txt")){          //如果此文件为可写模式
                if(fwrite($open,"今天是美好的一天，一定要开心哦！")>0){//当文件写入成功时
                    fclose($open);                  //关闭文件
                    echo "<script>alert('写入成功')</script>";         //输出成功提示
                }
            }
    }else{                                          //如果txt文件目录存在
        if(is_file("txt/in.txt")){                  //判断目录下是否存在in.txt文件
            if(is_readable("txt/in.txt")){          //判断此文件是否可读
                echo file_get_contents("txt/in.txt");              //输出文本数据信息
                unlink("txt/in.txt");               //删除in.txt文件
                rmdir('txt');                       //删除目录
            }
        }
    }
?>
```



运行结果如图 3.15 和图 3.16 所示。



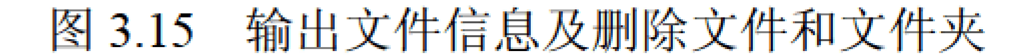


图 3.15　输出文件信息及删除文件和文件夹



图 3.16　创建文件夹及文件并写入信息

上机演练

上机演练 8　上传文件到服务器

在开发网站的过程中，不但可以将文件上传到数据库中，而且可以将文件上传到服务器中，这种方式可以为网站中的数据库节省很多空间，同时可以控制上传文件的大小、格式，并且读取服务器中的文件要比从数据库中读取方便很多。将文件上传到服务器中的具体操作为：单击图 3.17 中的“浏览”按钮，选择要上传的文件，然后单击“提交”按钮即可。如果上传成功，则在页面中提示“上传成功！”，否则给出错误提示信息。



图 3.17　上传文件

上机演练 9　读取整个文本文件的内容

在开发网站的过程中，很多的服务条款、协议等都是以文本文件形式存储的。如果要读取这些文件中的内容时就需要应用到文件系统函数，只有通过文件系统函数才能读取其中的内容。下面通过文件系统函数读取整个文件中的内容，其运行效果如图 3.18 所示。

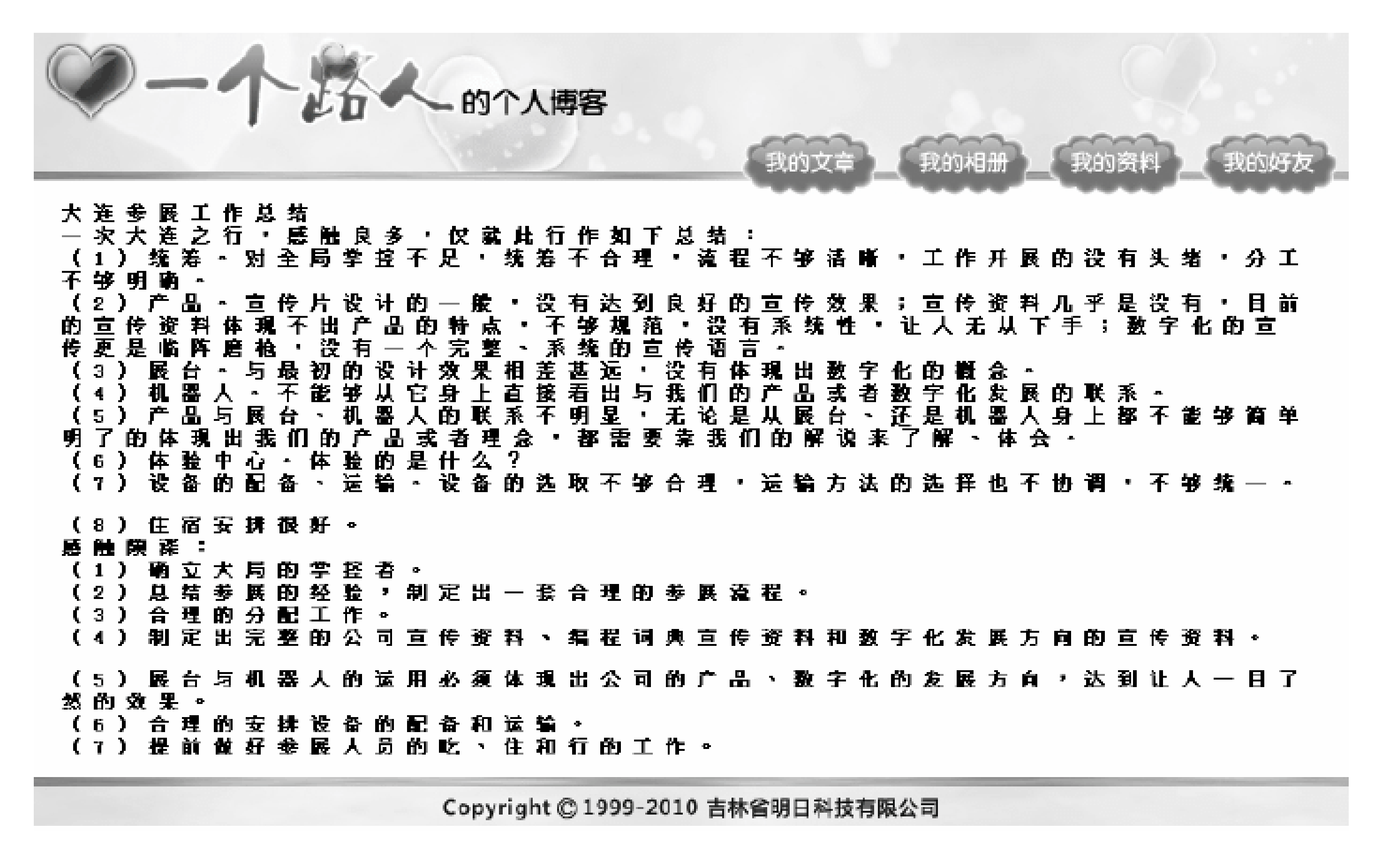
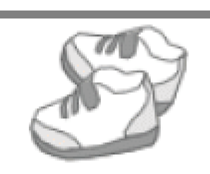
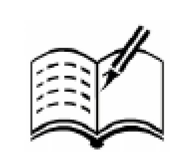


图 3.18　读取文本文件的内容





3.7 MySQL 函数库

视频讲解：配套资源\mr\3\video\MySQL 函数库.exe

PHP 支持多种数据库，而且内置了很多数据库操作的函数库。其中 MySQL 函数库是最为常用的一种。表 3.7 中列举了 MySQL 函数库中的常用函数。

表 3.7 常用的 MySQL 函数

函 数	功 能
mysql_close()	关闭 MySQL 连接
mysql_connect()	打开一个到 MySQL 服务器的连接
mysql_create_db()	新建一个 MySQL 数据库
mysql_error()	返回上一个 MySQL 操作产生的文本错误信息
mysql_fetch_array()	从结果集中获取一行作为关联数组或数字数组，或两者兼有
mysql_fetch_assoc()	从结果集中获取一行作为关联数组
mysql_fetch_field()	从结果集中获取列信息并作为对象返回
mysql_fetch_object()	从结果集中获取一行作为对象
mysql_fetch_row()	从结果集中获取一行作为枚举数组
mysql_num_rows()	获取结果集中行的数目
mysql_query()	发送一条 MySQL 查询
mysql_select_db()	选择 MySQL 数据库

上机演练

上机演练 10 连接 MySQL 数据库服务器

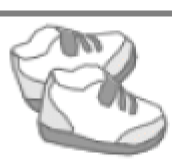
通过 PHP 语言连接 MySQL 数据库与在 cmd 命令提示符下操作数据库有所不同，例如设置页面的编码格式为“GBK”，在 MySQL 命令提示符下只需输入 set names gbk 即可，而在 PHP 中需使用函数“mysql_query('set names GBK')”实现，所以用户在学习操作 MySQL 数据库的同时还要与 PHP 函数联合应用，下面笔者为用户讲解 PHP 操作 MySQL 数据库的第一课，通过 mysql_connect()函数连接 MySQL 服务器，运行结果如图 3.19 所示。



图 3.19 通过 mysql_connect()函数连接 MySQL 服务器

上机演练 11 选择指定的 MySQL 数据库

通过 mysql_connect()函数可以与 MySQL 服务器建立连接，然后就可以操作数据库，在 cmd



命令提示符下选择想要操作的数据库需要使用 use 语句。在 PHP 代码中需要使用 mysql_select_db()函数选择并连接数据库。下面通过函数 mysql_select_db()实现与 MySQL 数据库的选择与连接，运行结果如图 3.20 所示。

Note

上机演练 12　执行 SQL 语句

用 PHP 代码操作 MySQL 数据库的目的是让程序控制数据库信息的增删改查等操作。而函数 mysql_query()可以看作是 SQL 语句的载体，SQL 语句需要通过 mysql_query()函数才能够执行。下面通过 mysql_query()函数执行 SQL 语句，运行结果如图 3.21 所示。



图 3.20　使用函数选择并连接 MySQL 数据库

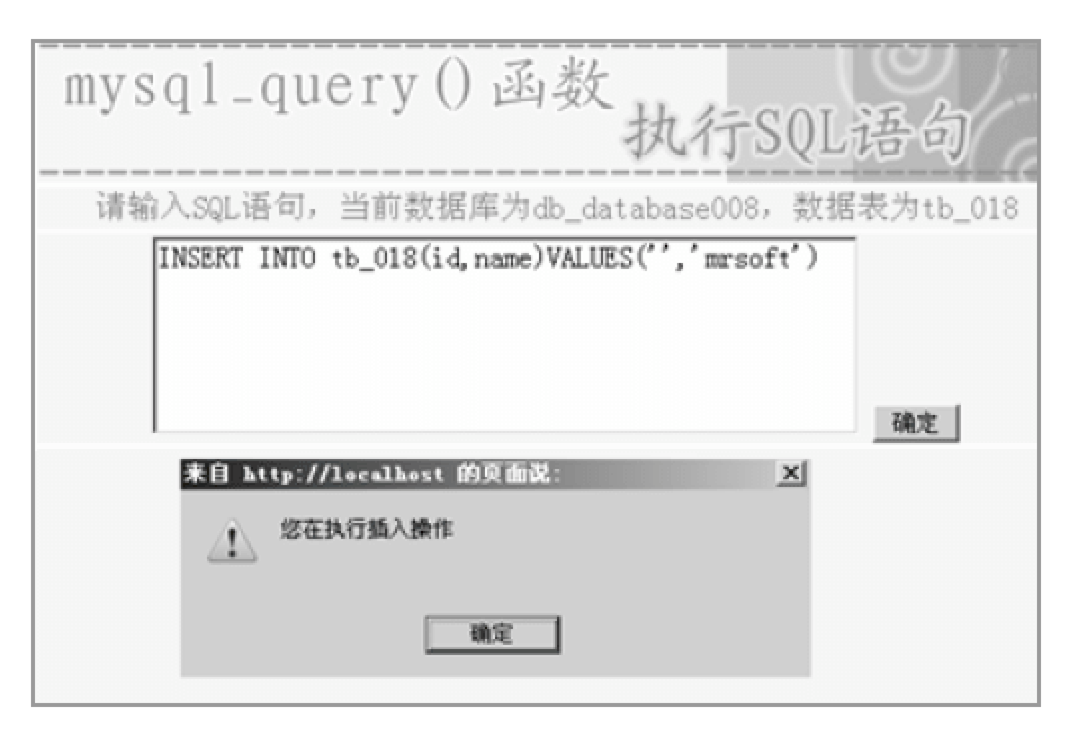


图 3.21　通过 mysql_query()函数执行 SQL 语句

本 章 摘 要

1．PHP 函数的创建和应用。
2．PHP 常用变量函数介绍。
3．PHP 常用字符串函数一览。
4．PHP 常用日期时间函数讲解。
5．PHP 常用数学函数介绍。
6．PHP 常用文件操作函数一览。
7．PHP 常用 MySQL 数据库操作函数介绍。

习　　题

1．下列哪个函数是将数组转换为字符串（　　）。
A．imploed()　　B．explode()
C．arsort()　　D．natsort()
2．下列哪个选项是正确引用文件的方法（　　）。
A．require 和 include　　B．require 和 function
C．define 和 include　　D．function 和 include
3．File()函数返回的数据类型为（　　）。
A．数组　　B．字符串





C．整型　　　　　　　　　　D．根据文件来定

4．通过（　　）函数打开或创建一个文件。

A．open()　　　B．fopen()　　　C．fwrite()　　　D. write()

5．以下字符的长度是（　　）。

```
<?php
    $text="   \tllo   ";
    echo strlen(trim($text));
?>
```

A．9　　　B．5　　　C．7　　　D．3

6．输出字符串可以应用函数（　　　）和（　　　）来输出。

7．PHP 中用来删除当前目录的函数是（　　　）。

8．（　　　）函数不能返回 UNIX 时间戳。

9．函数 vat_dump 的意义是：（　　　）。

10．下面代码的运行结果为（　　　）。

```
<?php
    $a="php china php mrkj";
    echo strrchr($a,"1")."<br>";
?>
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，同时给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 2 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　文本文件分页读取

在遍历文件中的内容时，由于文件内容很大，所以最理想的方法就是分页读取文本文件中的内容。通过自定义函数完成文本文件中数据的分页读取，其运行结果如图 3.22 所示。



图 3.22　文本文件的分页读取

实战模拟 2　查询关键字描红

关键字描红技术广泛应用于站内搜索或高级搜索中，是一项很常用的技术。下面通过



str_replace()函数实现查询关键字描红，运行结果如图 3.23 所示。

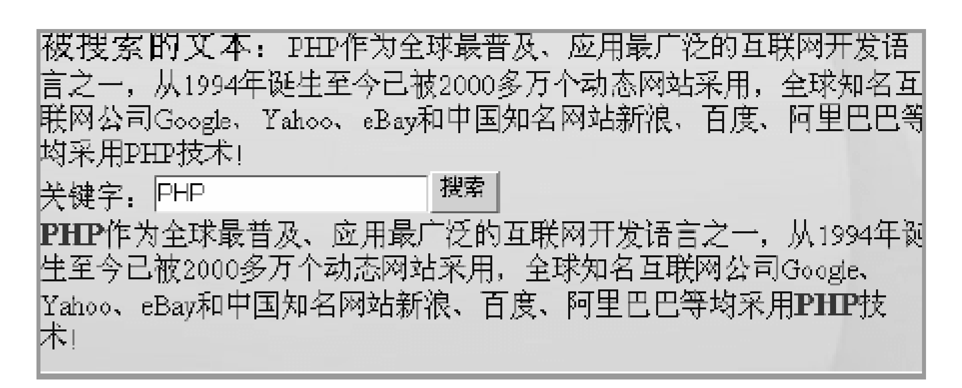


图 3.23　查询关键字描红

实战模拟 3　分页显示图书信息

利用 SELECT 语句可以查询数据库信息，但是如果数据库信息过多，则在一个页面中显示所有数据，首先会使页面显得臃肿、不美观，其次会给用户造成网站不够智能的假象。这里将灵活运用 SELECT 语句实现对图书信息的分页处理，其运行结果如图 3.24 所示。



ID	书名	价格	日期
6	《C#开发实战宝典》	75元	2010-07-16
7	《C++开发实战宝典》	75元	2010-07-16
8	《PHP范例宝典》	75元	2010-07-16
9	《VB范例宝典》	75元	2010-07-16
10	《JAVA范例宝典》	75元	2010-07-16

图 3.24　分页显示图书信息



PHP 流程控制语句

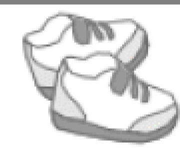
（自学视频、源程序：配套资源\mr\4\）

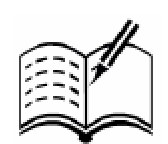
条件控制语句和循环控制语句是两种基本的语法结构，它们都是用来控制程序执行流程的，也是构成程序的主要语法基础。通常，语句是按顺序执行的，即执行完当前的语句之后，执行下一条语句。但是这种简单的顺序执行方式能够解决的问题十分有限，现实生活中的许多逻辑使用顺序执行的方式是不能够被描述的。例如，“如果明天不下雨，我们就去放风筝，否则我们呆在家里练习书法”。在程序中需要根据条件判断来确定是“放风筝”还是“呆在家里练习书法”，两种结果不可能同时存在，这时只能应用条件控制语句实现这个功能，而循环控制语句不能确定选择哪种结果。因此，开发人员可以按照实际问题的逻辑思路选择性地使用流程控制语句来编写程序代码。

学习摘要：

- 程序的 3 种控制结构
- if、switch 条件控制语句
- while、do…while 循环控制语句
- for、foreach 循环控制语句
- break、continue 跳转语句
- include、require 包含语句
- include_once、require_once 包含语句







4.1 程序的 3 种控制结构

在编程的过程中，所有的操作都是在按照某种结构有条不紊地进行，学习 PHP 语言，不仅要掌握其中的函数、数组和字符串等实际知识，更重要的是通过这些知识形成一种属于自己的编程思想和编程方法。要想形成属于自己的编程思想和方法，首先就要掌握程序设计的结构，再配合函数、数组和字符串等实际知识，逐步形成一种属于自己的编程方法。

程序设计的结构大致可以分为 3 种：顺序结构、选择结构和循环结构。在对这 3 种结构的使用中，几乎很少有哪个程序是单独使用某一种结构来完成某个操作的，基本上都是其中的两种或 3 种结构结合使用。

4.1.1 顺序结构

顺序结构是最基本的结构方式，各流程依次按顺序执行。传统流程图的表示方式与 N-S 结构化流程图的表示方式分别如图 4.1 和图 4.2 所示。执行顺序为：开始→语句 1→语句 2→…→结束。

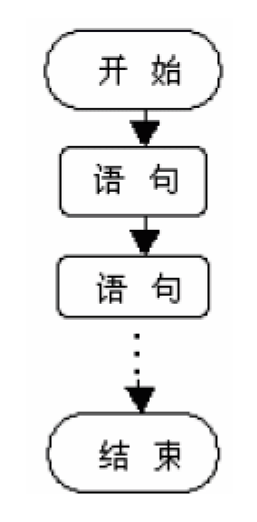


图 4.1 顺序结构传统流程图



图 4.2 N-S 结构化流程图

4.1.2 选择（分支）结构

选择结构就是对给定条件进行判断，当条件为真时执行一个分支，条件为假时执行另一个分支。其传统流程图表示方式与 N-S 结构化流程图表示方式分别如图 4.3 和图 4.4 所示。

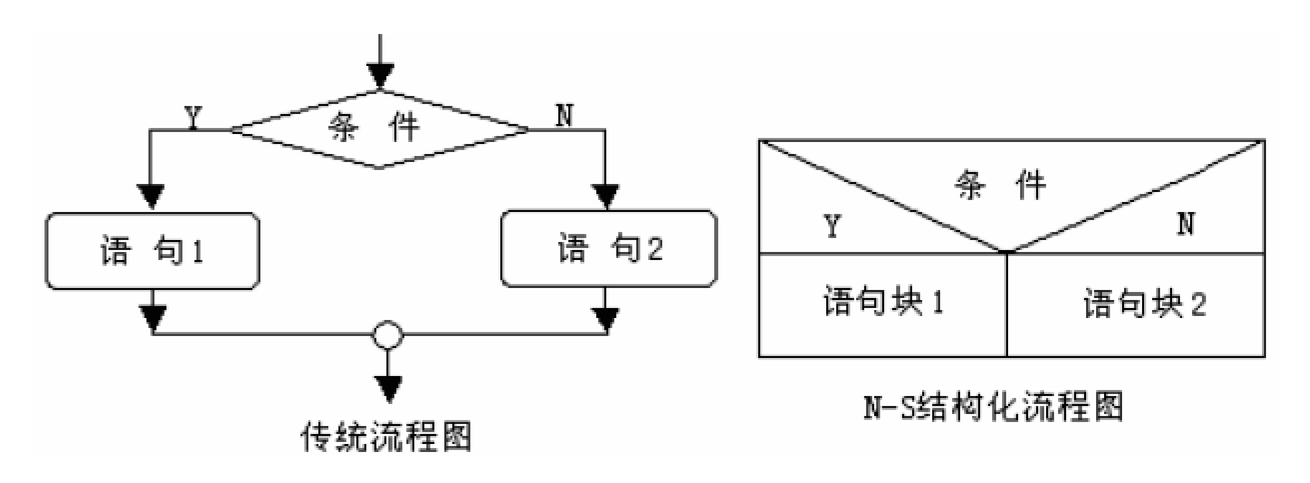


图 4.3 条件成立与否都执行语句或语句块

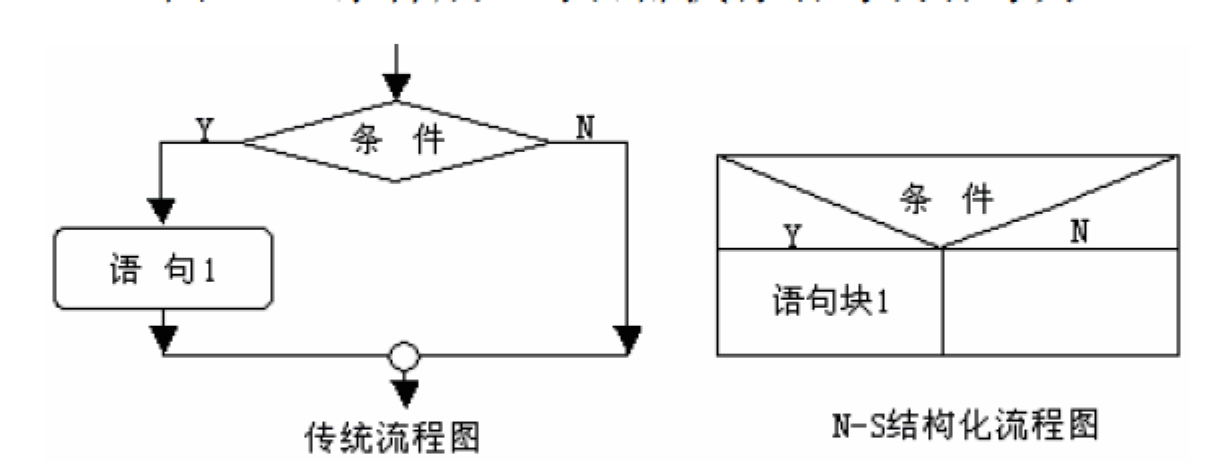
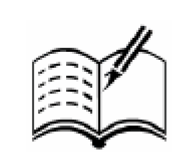


图 4.4 条件为否不执行语句或语句块





Note

4.1.3　循环结构

循环结构可以按照需要多次重复执行一行或多行代码。该结构分为两种：前测试型循环和后测试型循环。

（1）前测试型循环，先判断后执行。当条件为真时反复执行语句或语句块，条件为假时，跳出循环，继续执行循环后面的语句。流程图如图 4.5 所示。

（2）后测试型循环，先执行后判断。先执行语句或语句块，再进行条件判断，直到条件为假时，跳出循环，继续执行循环后面的语句，否则将一直执行语句或语句块。流程图如图 4.6 所示。

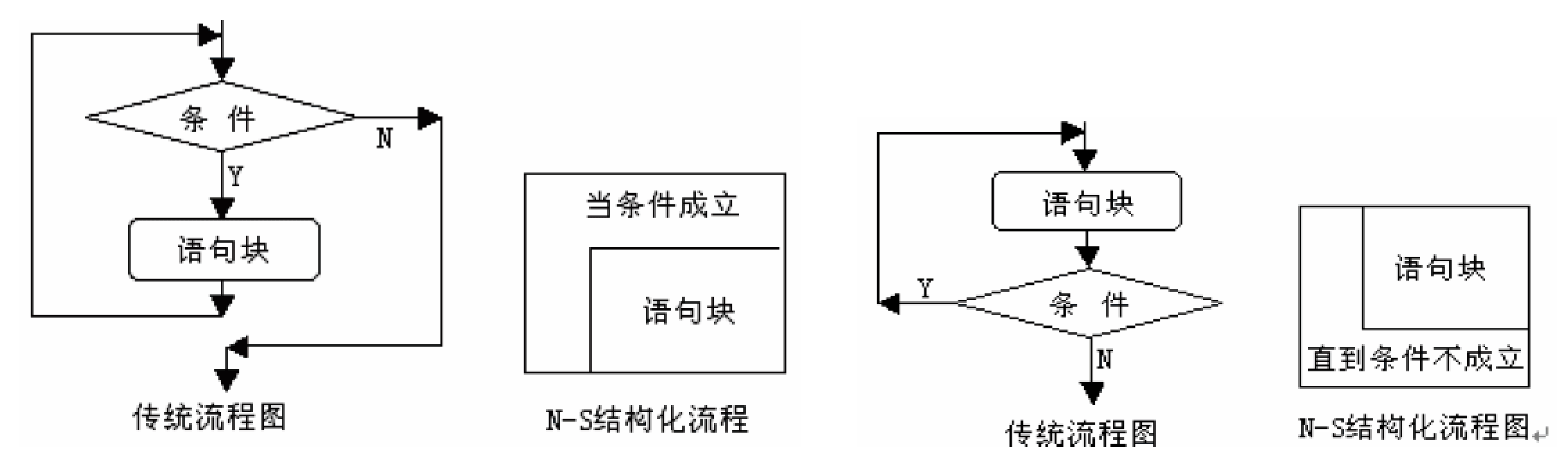


图 4.5　前测试型循环流程图　　　图 4.6　后测试型循环流程图

在大多数情况下，PHP 中的程序都以这 3 种结构的组合形式出现。其中的顺序结构很容易理解，就是直接输出程序运行结果，而选择和循环结构则需要一些特殊的控制语句来实现，包括以下 3 种控制语句。

- ☑ 条件控制语句：if、else、elseif 和 switch。
- ☑ 循环控制语句：while、do…while、for 和 foreach。
- ☑ 跳转控制语句：break、continue 和 return。

4.2　条件控制语句

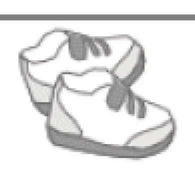
视频讲解：配套资源\mr\4\video\条件控制语句.exe

所谓条件控制语句就是对语句中不同条件的值进行判断，进而根据不同的条件执行不同的语句。在条件控制语句中主要有两个语句：if 条件控制语句和 switch 多分支语句。

4.2.1　if 条件控制语句

if 条件控制语句是所有流程控制语句中最简单、最常用的一个，根据获取的不同条件判断执行不同的语句。该语句应用范围十分广泛，无论程序大小几乎都会应用到它。其语法如下：

```
if (expr)
    statement;                //这是基本的表达式
if () {}                      //这是执行多条语句的表达式
```



```
if () {}else {}                //这是通过else延伸了的表达式
if () {}elseif() {} else {}    //这是加入了elseif同时判断多个条件的表达式
```

参数 expr 按照布尔求值。如果 expr 的值为 True，则执行 statement；如果 expr 的值为 False，则忽略 statement。if 语句可以无限层地嵌套到其他 if 语句中，实现更多条件的执行。

else 的功能是当 if 语句在参数 expr 的值为 False 时执行其他语句，即在执行的语句不满足该条件时执行 else 后大括号中的语句。

在同时判断多个条件时，PHP 提供了 elseif 的语句来扩展需求。elseif 语句被放置在 if 和 else 语句之间，以满足多条件同时判断的需求。

if 语句的流程如图 4.7、图 4.8 和图 4.9 所示。

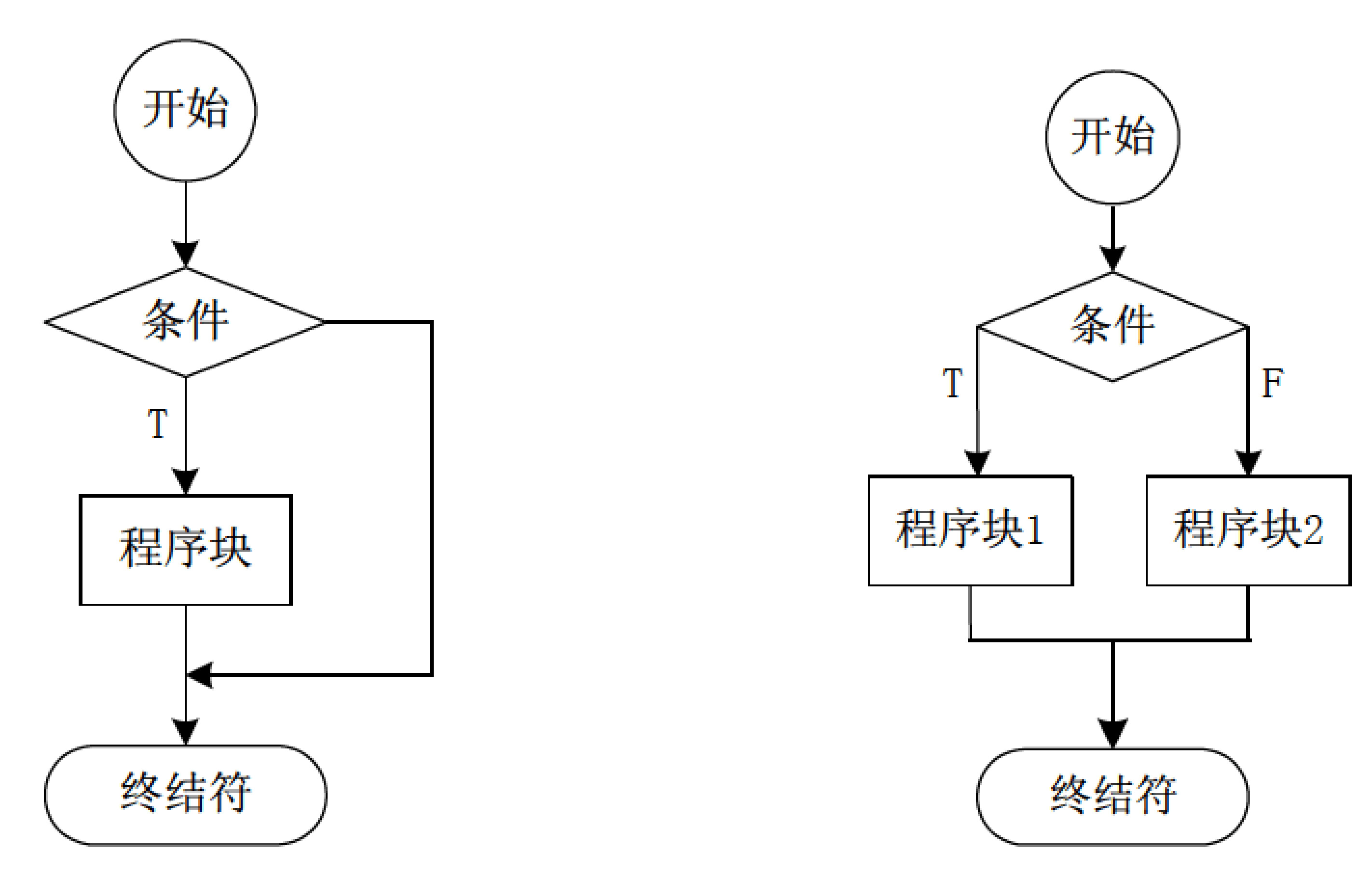


图 4.7　if 语句流程图　　图 4.8　if...else 语句流程控制图

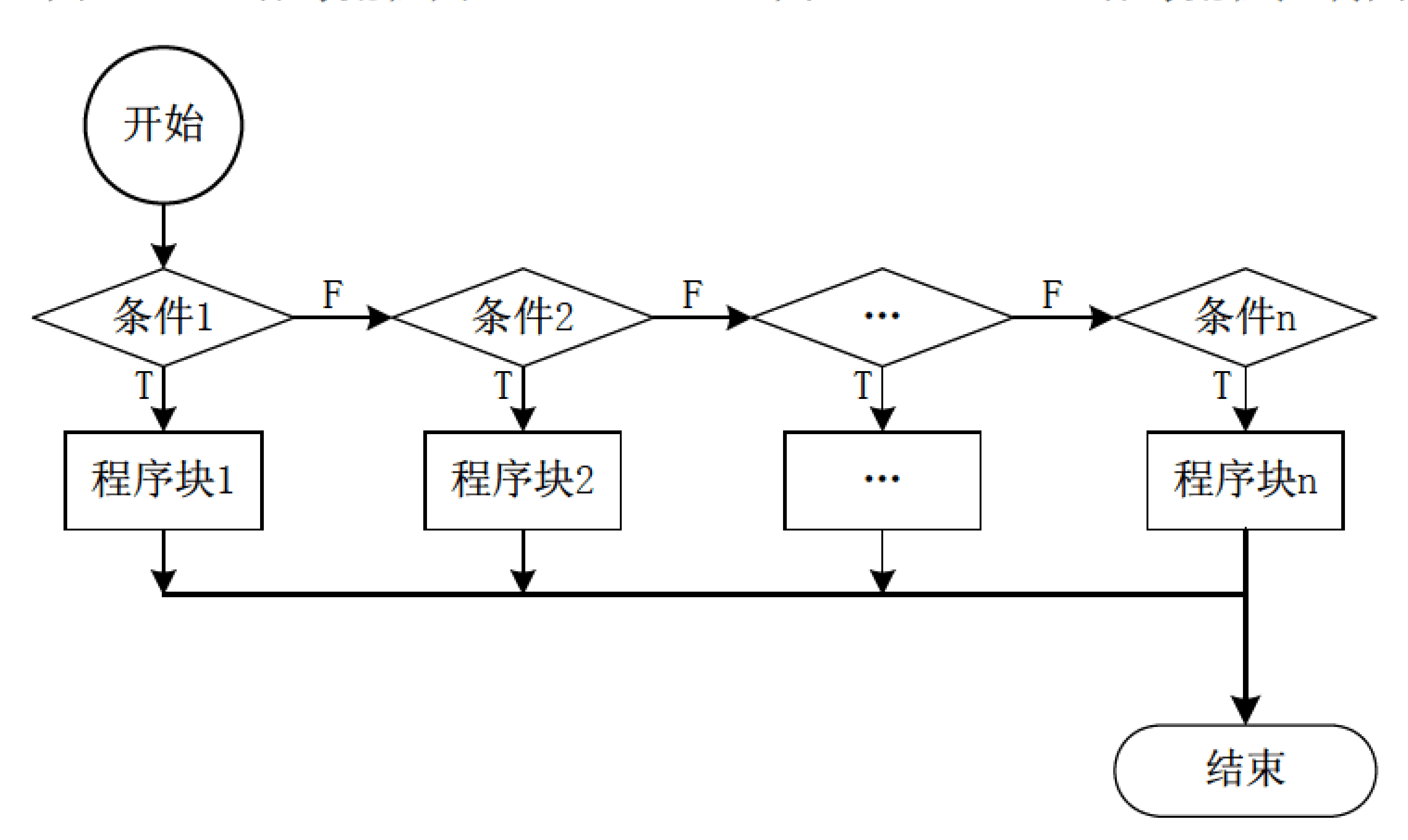


图 4.9　elseif 语句的流程控制图

例 4.1　通过 if 语句判断用户提交的登录信息是否为空。（**实例位置：配套资源\mr\4\example\4.1**）

创建 index.php 文件，创建一个用户登录页面，提交登录的用户名和密码。然后，在页面中通过$_POST[]方法获取表单中提交的用户名和密码，并且应用 if 语句判断用户提交的登录信息是否为空。关键代码如下：

```
<?php
/*
$_POST[]方法获取表单提交的按钮“sub”、用户名“textt”和密码“pwd”的值
```





```
isset()函数检测按钮变量sub是否存在。如果存在返回True，否则返回False
$_POST[text]!=""||$_POST[pwd]!=""判断用户名和密码是否为空，不为空返回true，否则返回False
测试成功通过echo语句返回提示信息
*/
<?php
    if(isset($_POST['sub'])){                              //判断提交按钮值是否存在
        if($_POST['text']!="" && $_POST['pwd']!=""){       //判断提交的数据是否为空
            echo "<script>alert('测试成功');</script>";
        }else{
            echo "<script>alert('文本框内容不能为空');</script>";
        }
    }
?>
```

运行结果如图 4.10 所示。

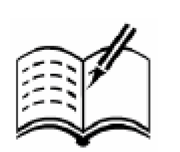


图 4.10　用户登录模块的实现

4.2.2　switch 多分支语句

switch 语句和 if 条件控制语句类似，实现将同一个表达式与很多不同的值比较，获取相同的值，并且执行相同的值对应的语句。其语法如下：

```
<?php
switch ( expr ){                  //expr条件为变量名称
        case expr1:               //case后的expr1为变量的值
            statement1;           //冒号":"后的是符合该条件时要执行的部分
          break ;                 //应用break跳离循环体
        case expr2 :
            statement2 ;
          break ;
      default:
          statementN;
          break;
}
?>
```


Note

switch 语句的参数说明如表 4.1 所示。

表 4.1　switch 语句的参数说明

参　数	说　明
expr	表达式的值，即 switch 语句的条件变量的名称
expr1	放置于 case 语句之后，是要与条件变量 expr 进行匹配的值中的一个
statement1	在参数 expr1 的值与条件变量 expr 的值相匹配时执行的代码
break 语句	终止语句的执行，即当语句在执行过程中，遇到 break 就停止执行，跳出循环体
default	case 的一个特例，匹配任何其他 case 都不匹配的情况，并且是最后一条 case 语句

switch 语句的流程控制如图 4.11 所示。

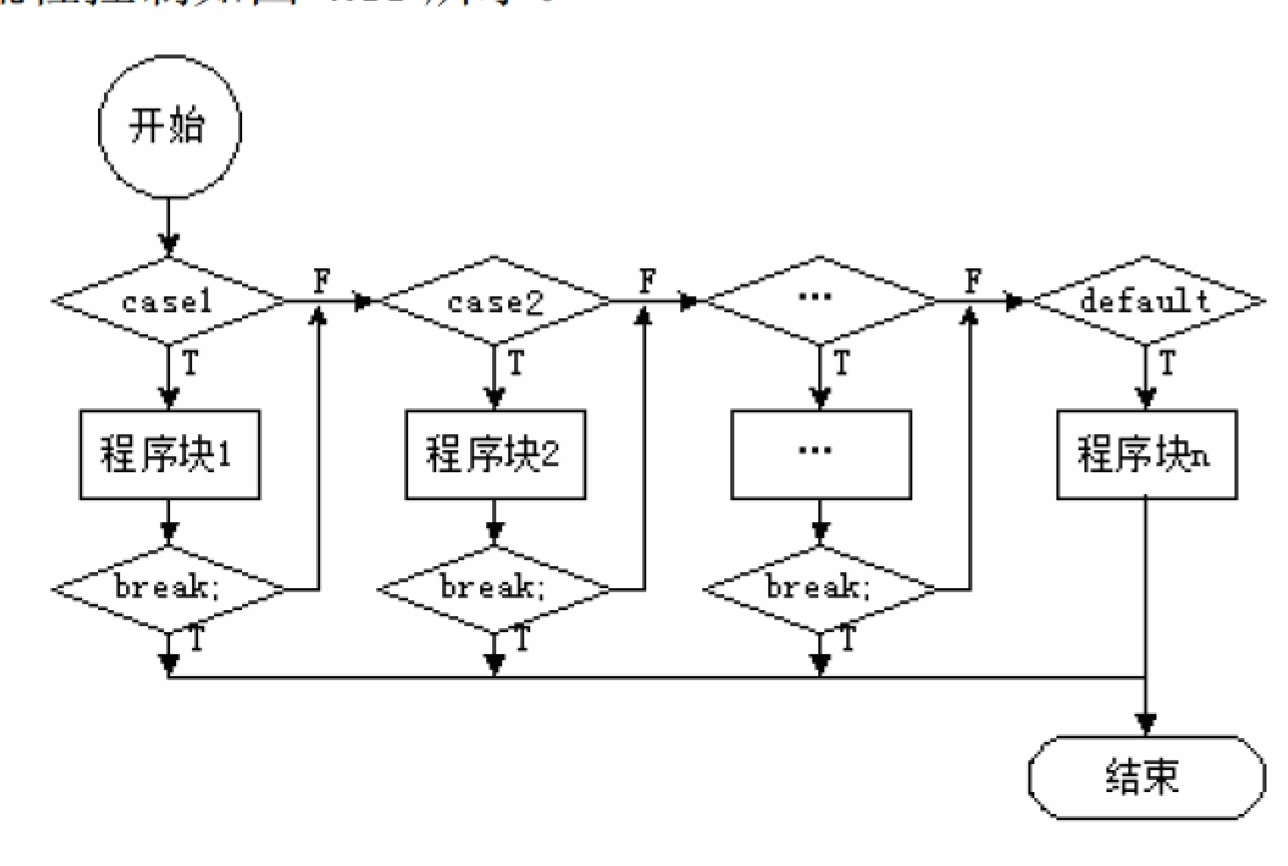
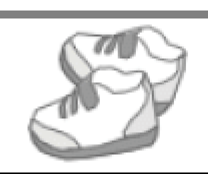


图 4.11　switch 语句流程控制图

例 4.2　应用 switch 语句判断成绩的等级情况，代码如下：（**实例位置：配套资源\mr\4\example\4.2**）

```
<?php
    $cont=49;                  //以下代码实现了根据$cont的值，判断成绩等级的功能
    switch($cont) {
        case $cont==100;       //如果$cont的值等于100，则输出“满分”
            echo"满分";
            break;
        case $cont>=90;        //如果$cont的值大于等于90，则输出“优秀”
            echo"优秀";
            break;
        case $cont>=60;        //如果$cont的值大于等于60，则输出“及格”
            echo"及格";
            break;
        default:               //如果$cont的值小于60，则输出“不及格”
            echo"不及格";
    }
?>
```

运行结果为：不及格





指点迷津：

if 和 switch 语句可以从使用的效率上来进行区别，也可以从实用性角度去区分。如果从使用的效率上进行区分，在对同一个变量的不同值作条件判断时，使用 switch 语句的效率相对更高一些，尤其是判断的分支越多越明显。

如果从语句的实用性的角度去区分，则 switch 语句肯定不如 if 条件语句，可以说，if 条件语句是实用性最强、应用范围最广的语句。

在程序开发的过程中，if 和 switch 语句的使用应该根据实际的情况而定，不要因为 switch 语句的效率高就一味地使用，也不要因为 if 语句常用就不使用 switch 语句，要根据实际情况，具体问题具体分析，使用最适合的条件语句。一般情况下可以使用 if 条件语句，但是在实现一些多条件的判断中，特别是在实现框架的功能时就应使用 switch 语句。

上机演练

上机演练 1　员工生日提醒

在 PHP 应用中类似生日提醒等功能的程序随处可见，主要是因为这样的程序有一定的定时效果。通过 if 语句和 foreach 循环语句实现一个员工生日提醒的小程序，如图 4.12 所示。

上机演练 2　switch 网页框架

类似留言板等功能较单一的网站，可以通过 switch 语句制作网页框架将所有内容包含到主页中。下面通过 switch()和 include()语句来演示网页框架的制作，运行结果如图 4.13 所示。



图 4.12　员工生日提醒

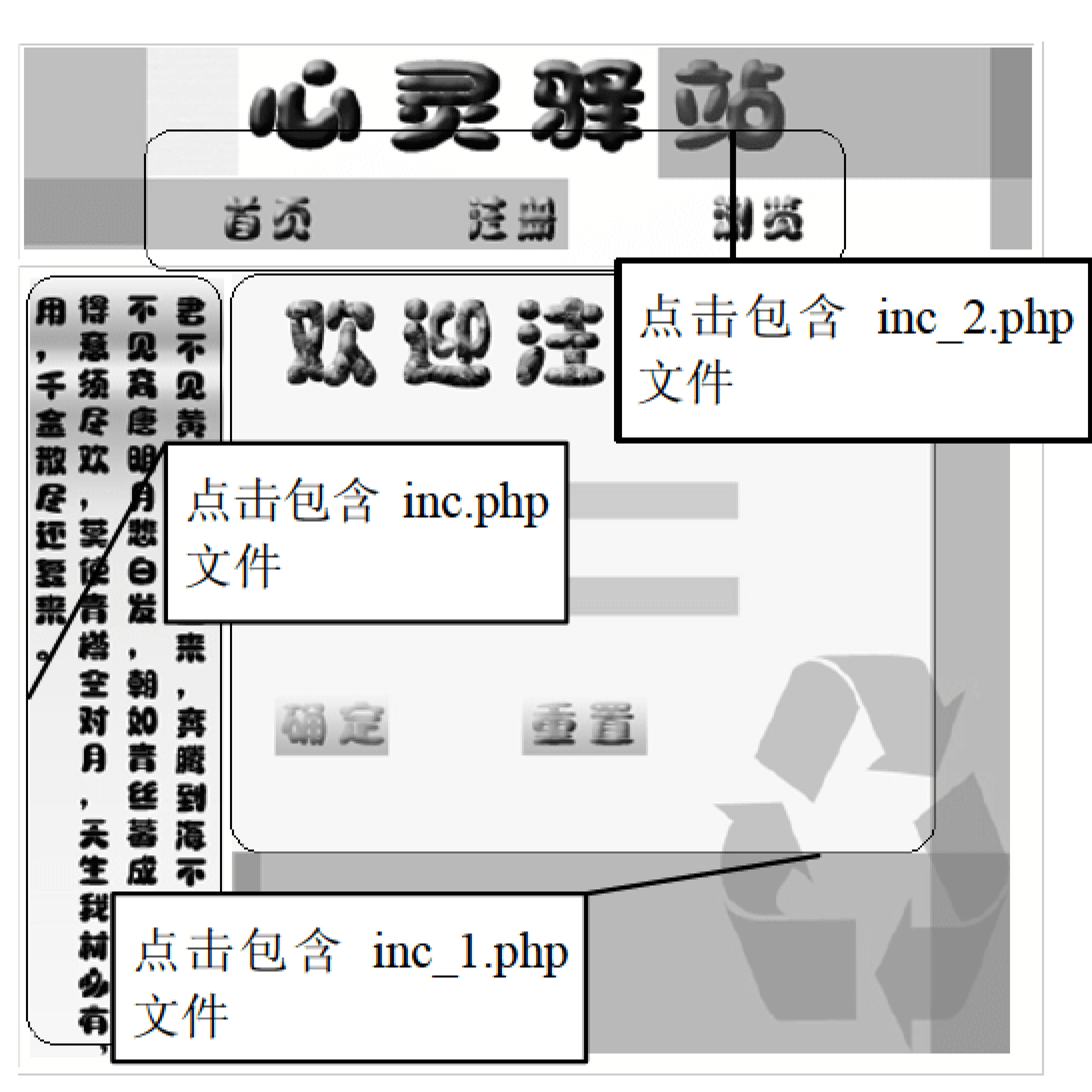
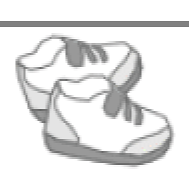


图 4.13　网页框架的制作

4.3　循环控制语句

视频讲解：配套资源\mr\4\video\循环控制语句.exe

循环语句是在满足条件的情况下反复地执行某一个操作。在 PHP 中，提供 4 个循环控制





语句，分别是：while 循环语句、do…while 循环语句、for 循环语句和 foreach 循环语句。

4.3.1 while 循环语句

while 循环语句，其作用是反复地执行某一项操作，是循环控制语句中最简单的一个，也是最常用的一个。while 循环语句对表达式的值进行判断，当表达式为非 0 值时，执行 while 语句中的内嵌语句；当表达式的值为 0 值时，不执行 while 语句中的内嵌语句。该语句的特点是：先判断表达式，后执行语句。while 循环控制语句的操作流程如图 4.14 所示。

其语法如下：

```
while (expr){                    /*
        statement;               先判断条件，当条件满足时执行语句块，否则
                                 不向下执行
}                                */
```

只要 while 表达式 expr 的值为 True，就重复执行嵌套中的 statement 语句；如果 while 表达式的值一开始就是 False，则循环语句一次也不执行。

例 4.3 展示 while 语句的应用，计算员工的工龄工资。正式员工在单位工作每增加一年，工龄工资增长 50，以 10 年为上限，计算一个员工总的工龄工资增加情况。核心代码如下：**（实例位置：配套资源\mr\4\example\4.3）**

```
<?php
$a=1;
$year=10;
while($a<=$year){
$price=50*12*$a;
        echo "您第".$a."年的工龄工资为<b>".$price."</b>元<br>";
        $a++;
}
?>
```

运行结果如图 4.15 所示。

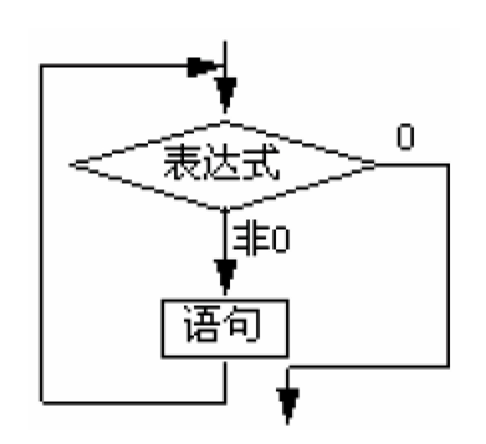
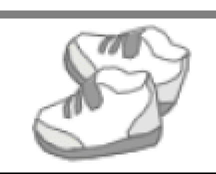


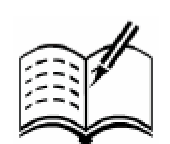
图 4.14 while 循环控制语句的操作流程

计算员工的工龄工资

您第1年的工龄工资为600元
您第2年的工龄工资为1200元
您第3年的工龄工资为1800元
您第4年的工龄工资为2400元
您第5年的工龄工资为3000元
您第6年的工龄工资为3600元
您第7年的工龄工资为4200元
您第8年的工龄工资为4800元
您第9年的工龄工资为5400元
您第10年的工龄工资为6000元

图 4.15 计算员工的工龄工资





脚下留神：

在应用 while 计算员工的工龄工资时，如果将变量 a 的值定义为 11，那么将不会输出员工第 11 年的工资。但是，如果应用下面的 do…while 循环语句执行此项操作时，就会得到意想不到的结果，而这个结果正是这两个语句之间区别的体现。

4.3.2　do...while 循环语句

do…while 语句也是循环控制语句中的一种，其使用方式和 while 相似，也是通过判断表达式的值来输出循环语句。其语法如下：

```
do{                        /*
                  程序在未经判断之前就进行了一次循环，循环到while部分才判断
        statement;         条件，即使条件不满足，程序也已经运行了一次
*/
}while(expr);
```

该语句的操作流程是：先执行一次指定的循环体语句，然后判断表达式的值，当表达式的值为非 0 时，返回重新执行循环体语句，如此反复，直到表达式的值等于 0 为止，此时循环结束。其特点是先执行循环体，然后判断循环条件是否成立。do…while 循环语句的操作流程如图 4.16 所示。

例 4.4　通过 do…while 语句计算一个员工总的工龄工资增加情况。核心代码如下：（**实例位置：配套资源\mr\4\example\4.4**）

```
<?php
    $a=1;               //定义变量$a的值为1
    $year=10;
    do{
        $price=50*12*$a;
        echo "您第".$a."年的工龄工资为<b>".$price."</b>元<br>";
        $a++;
    }while($a<=$year);
?>
```

运行结果如图 4.17 所示。

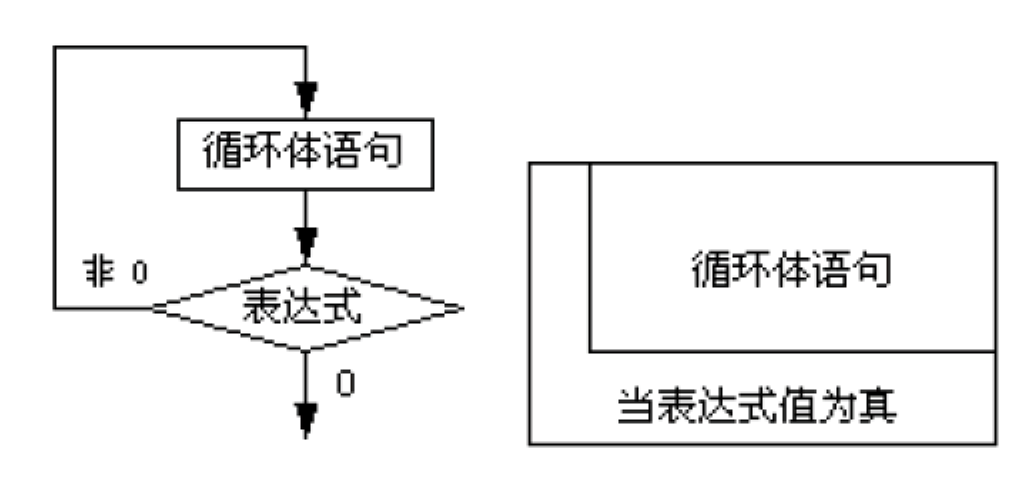
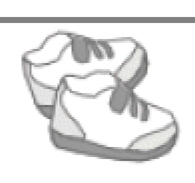


图 4.16　do…while 循环语句的操作流程

计算员工的工龄工资

您第1年的工龄工资为600元
您第2年的工龄工资为1200元
您第3年的工龄工资为1800元
您第4年的工龄工资为2400元
您第5年的工龄工资为3000元
您第6年的工龄工资为3600元
您第7年的工龄工资为4200元
您第8年的工龄工资为4800元
您第9年的工龄工资为5400元
您第10年的工龄工资为6000元

图 4.17　计算员工的工龄工资（一）





Note

前面已经讲过，如果使用 do…while 语句计算员工的工龄工资，当变量 a 的值等于 11 时，会得到一个意想不到的结果。下面就来实践操作，看具体会得到一个什么样的结果。定义变量 a 的值为 11，重新执行示例，其代码如下。

```
<?php
    $a=11;              //当直接定义变量$a的值为11时，仍可以输出第11年的工资
    $year=10;           //定义初始变量$year=10
    do{
        $price=50*12*$a;
        echo "您第".$a."年的工龄工资为<b>".$price."</b>元<br>";
        $a++;
    }while($a<=$year); //当$year等于10时程序没有停止，继续计算第11年工资；当$year等于11时判断条件不符合停止循环，但是第11年的工资已经输出了
?>
```

运行结果如图 4.18 所示。

计算员工的工龄工资

您第11年的工龄工资为6600元

图 4.18　计算员工的工龄工资（二）

指点迷津：

以上就是 while 和 do…while 语句之间的区别。do…while 语句是先执行后判断，无论表达式的值是否为 True，都将执行一次循环；而 while 语句则是首先判断表达式的值是否为 True，如果为 True 则执行循环语句，否则将不执行循环语句。

编写这个示例意在说明 while 语句与 do…while 语句在执行判断上的一个小小区别，在实际的程序开发中不会出现上述情况。

4.3.3　for 循环语句

for 语句是 PHP 中最复杂的循环控制语句，拥有 3 个条件表达式。其语法如下：

```
for (expr1; expr2; expr3){
    statement
}
```

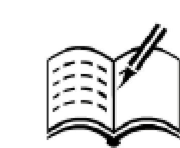
for 循环语句的参数说明如表 4.2 所示。

表 4.2　for 循环语句的参数说明

参　　数	说　　明
expr1	必选参数。第 1 个条件表达式，在第一次循环开始时被执行
expr2	必选参数。第 2 个条件表达式，在每次循环开始时被执行，决定循环是否继续
expr3	必选参数。第 3 个条件表达式，在每次循环结束时被执行
statement	必选参数。满足条件后，循环执行的语句

其执行过程为：首先执行表达式 1，然后执行表达式 2，并对表达式 2 的值进行判断。如果值为真，则执行 for 循环语句中指定的内嵌语句；如果值为假，则结束循环，跳出 for 循环





语句。最后执行表达式 3（切忌是在表达式 2 的值为真时），返回表达式 2 继续循环执行。for 循环语句的操作流程如图 4.19 所示。

例 4.5　使用 for 循环来计算 2～100 之间所有偶数之和。核心代码如下：（**实例位置：配套资源\mr\4\example\4.5**）

```
<?php
    $b="";
    for($a=0;$a<=100;$a+=2){          //执行for循环
        $b=$a+$b;                      //计算所有偶数之和
    }
    echo "结果为：<b>".$b."</b>";
?>
```

运行结果如图 4.20 所示。

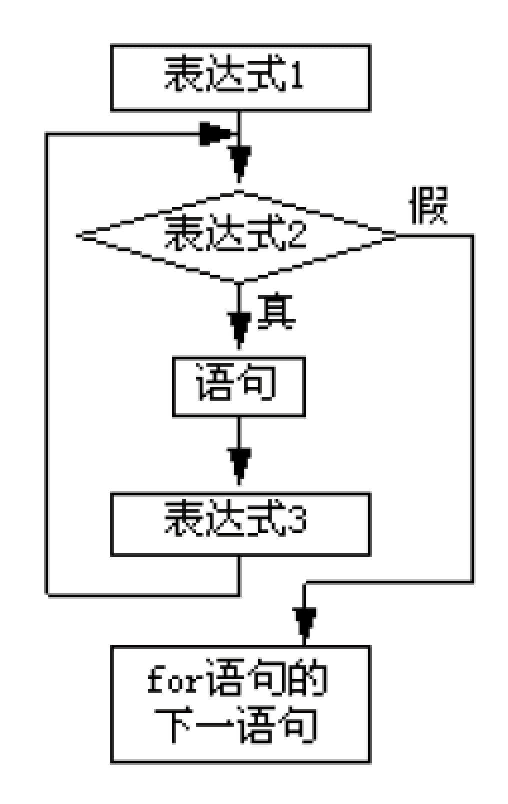


图 4.19　for 循环语句的操作流程图

计算2~100所有偶数的和
结果为：2550

图 4.20　计算 2～100 之间所有偶数之和

多学两招：

在编程时，有时会遇到使用 for 循环的特殊语法格式来实现无限循环。语法格式为：

```
for(;;){
   …
}
```

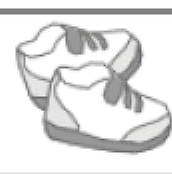
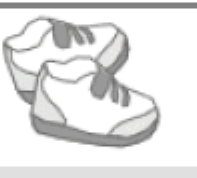
对于这种无限循环可以通过 break 语句跳出循环。例如：

```
for(;;){
if(x <20)
        break;
    x++;
}
```

4.3.4　foreach 循环语句

foreach 循环控制语句自 PHP4 开始被引入，主要用于处理数组，是遍历数组的一种简单方法。如果将该语句用于处理其他的数据类型或初始化的变量，将会产生错误。该语句的语法有两种格式：

```
foreach (array_expression as $value){
    statement
```

```
}
foreach (array_expression as $key => $value){
    statement
}
```

参数 array_expression 是指定要遍历的数组，其中$value 是数组的值，$key 是数组的键名，statement 是满足条件时要循环执行的语句。

在第一种格式中，当遍历指定的 array_expression 数组时，每次循环时，将当前数组单元的值赋给变量$value，并且将数组中的指针移动到下一个单元。

在第二种格式中的应用是相同的，只是在将当前单元的值赋给变量$value 的同时，将当前单元的键名也赋给了变量$key。

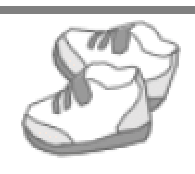
多学两招：

当使用 foreach 语句用于其他数据类型或未初始化的变量时会产生错误。为了避免出现这个问题，最好使用 is_array()函数先来判断变量是否为数组类型，如果是，再进行其他操作。

例 4.6 应用 foreach 语句处理一个数组，实现输出购物车中商品的功能。这里假设将购物车中的商品存储于指定的数组中，然后通过 foreach 语句来输出购物车中的商品信息，其关键代码如下：（**实例位置：配套资源\mr\4\example\4.6**）

```
<?php
/*
    PHP中的数组元素较其他编程语言有所不同，PHP中的数组下标可以为数字，默认情况下数组下标以0为开始，数组下标还可以使用字符串作为数组键值，具体的细节可参考本书第5章
*/
$name = array("1"=>"钢笔","2"=>"衬衫","3"=>"手机","4"=>"电脑");//定义数组并赋值
$price = array("1"=>"108元","2"=>"88元","3"=>"666元","4"=>"6666元");
$counts = array("1"=>1,"2"=>1,"3"=>2,"4"=>1);
echo '<table width="480" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#FF0000">
<tr>
    <td width="144" align="center" bgcolor="#FFFFFF"   class="STYLE1">商品名称</td>
    <td width="144" align="center" bgcolor="#FFFFFF"   class="STYLE1">价  格</td>
    <td width="144" align="center" bgcolor="#FFFFFF"   class="STYLE1">数量</td>
    <td width="144" align="center" bgcolor="#FFFFFF"   class="STYLE1">金额</td>
</tr>';
foreach($name as $key=>$value){                    //使用foreach语句遍历数组，输出键和值
    echo '<tr>
        <td height="24" align="center" bgcolor="#FFFFFF">'.$value.'</td>
        <td align="center" bgcolor="#FFFFFF">'.$price[$key].'</td>
        <td align="center" bgcolor="#FFFFFF">'.$counts[$key].'</td>
        <td align="center" bgcolor="#FFFFFF">'.$counts[$key]*$price[$key].'</td>
</tr>';
}
echo '</table>';
?>
```

运行结果如图 4.21 所示。





商品名称	价 格	数量	金额
钢笔	108元	1	108
衬衫	88元	1	88
手机	666元	2	1332
电脑	6666元	1	6666

图 4.21　应用 foreach 语句输出购物车中商品

上机演练

上机演练 3　员工信息批量删除

在操作数据库的过程中可能会出现很多无用的冗余信息，想要删除它们，用户可以使用 while 语句循环删除。通过 while 循环语句实现员工信息的批量删除，运行结果如图 4.22 所示。

上机演练 4　do…while 循环语句读取数据库中数据

通过 do…while 循环语句循环读取数据库中的数据，输出员工详细信息，运行结果如图 4.23 所示。

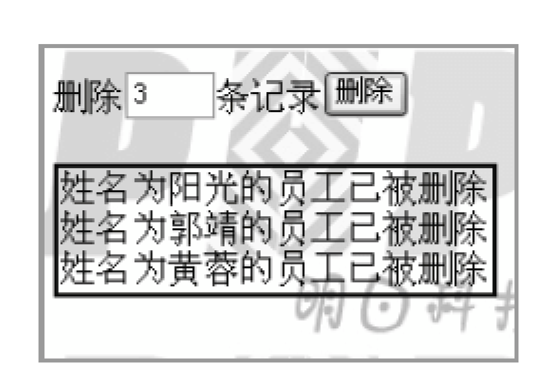


图 4.22　员工信息的批量删除

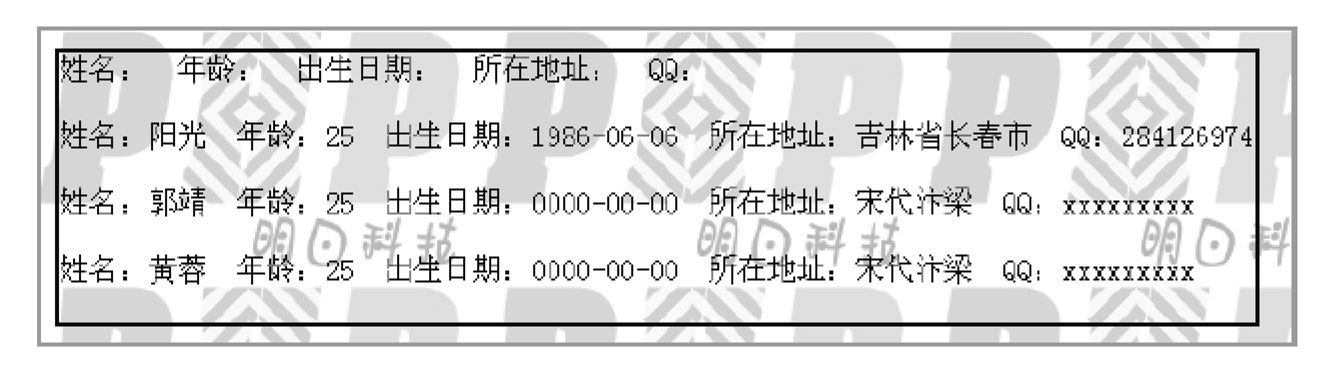


图 4.23　员工详细信息浏览

上机演练 5　生成随机验证码

在用户登录页面或留言发表页面经常会看到验证码，通过这项技术可以很大程度地提高网站的安全性。在如图 4.24 所示的登录页面中随机产生了 4 位数字（6191），每次刷新页面这 4 位数字都在发生改变，用户登录时要求输入这 4 位数字，这样可以防止有些非法用户通过恶意程序来试探登录密码的值，从而非法登录网站。



图 4.24　生成随机验证码



4.4 跳 转 语 句



Note

视频讲解：配套资源\mr\4\video\跳转语句.exe

跳转语句有 3 个：break 语句、continue 语句和 return 语句。其中前两个跳转语句使用起来非常简单而且非常容易掌握，主要原因是它们都被应用在指定的环境中，例如 for 循环语句中。return 语句在应用环境上较前两者相对单一，一般被用在自定义函数和面向对象的类中。

4.4.1 break 跳转语句

break 关键字可以终止当前的循环，包括 while、do…while、for、foreach 和 switch 在内的所有控制语句。

break 语句不仅可以跳出当前的循环，还可以指定跳出几重循环。格式为：

```
break    n;
```

参数 n 指定要跳出的循环数量。break 关键字的流程图如图 4.25 所示。

例 4.7 应用 for 循环控制语句声明变量$i，循环输出 4 个表情头像，当变量$i 等于 4 时，使用 break 跳转控制语句跳出 for 循环，代码如下：（**实例位置：配套资源\mr\4\example\4.7**）

```
<?php
for($i=1;$i<=4;$i++){                    //应用for循环控制语句输出表情头像
    if($i==4){                           //判断变量是否等于4
        break;                           //如果等于4，使用break语句跳转循环
    }
?>
<input type="radio" name="head" value="<?php echo("images/".$i.".jpg");?>" />
<img src="<?php echo("images/".$i.".jpg");?>" width="90" height="90" id="head"/>
<?php
   }
?>
```

运行结果如图 4.26 所示。

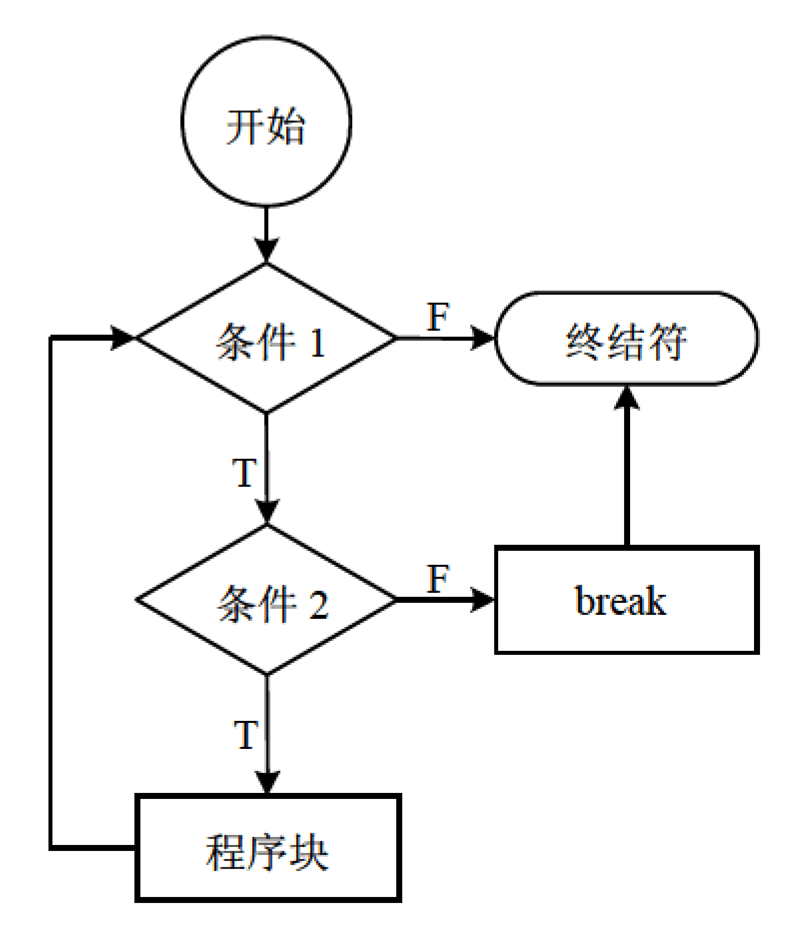
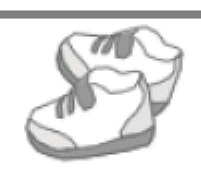


图 4.25 break 关键字的流程图



图 4.26 应用 break 跳转控制语句跳出循环





4.4.2　continue 跳转语句

程序执行 break 语句后将跳出循环，开始继续执行循环体的后续语句。而在执行 continue 语句后，程序将结束本次循环的执行，并开始执行下一轮循环或指定跳出几重循环。continue 跳转语句的流程图如图 4.27 所示。

例 4.8　使用 for 循环输出数组元素。如果数组下标为偶数，则只输出一个空行；如果是奇数，则继续输出。在最里面的循环中，判断当前数组下标是否等于$i，如果不相等，输出数组元素，否则跳到最外重循环。代码如下：（**实例位置：配套资源\mr\4\example\4.8**）

```
<?php
$arr = array("PHP程序开发范例宝典","PHP典型模块开发大全","PHP函数参考大全","PHP项目开发
全程实录", "PHP从入门到精通","PHP网络编程自学手册");          //声明一个数组变量$arr
for($i = 0; $i < 6; $i++){                              //使用for循环
    echo "<br>";
    if($i % 2 == 0){                                    //如果$i的值为偶数，则跳出本次循环
        continue;
    }
    for(;;){                                            //无限循环
        for($j = 0; $j < count($arr); $j++){            //再次使用for循环输出数组变量
            if($j == $i){                               //如果当前输出的数组下标等于$i
                continue 3;                             //跳出最外重循环
            }else{
                echo $arr[$j]."\t | ";                  //输出表达式
            }
        }
    }
echo "你永远都看不到我噢!";
}
?>
```

运行结果如图 4.28 所示。

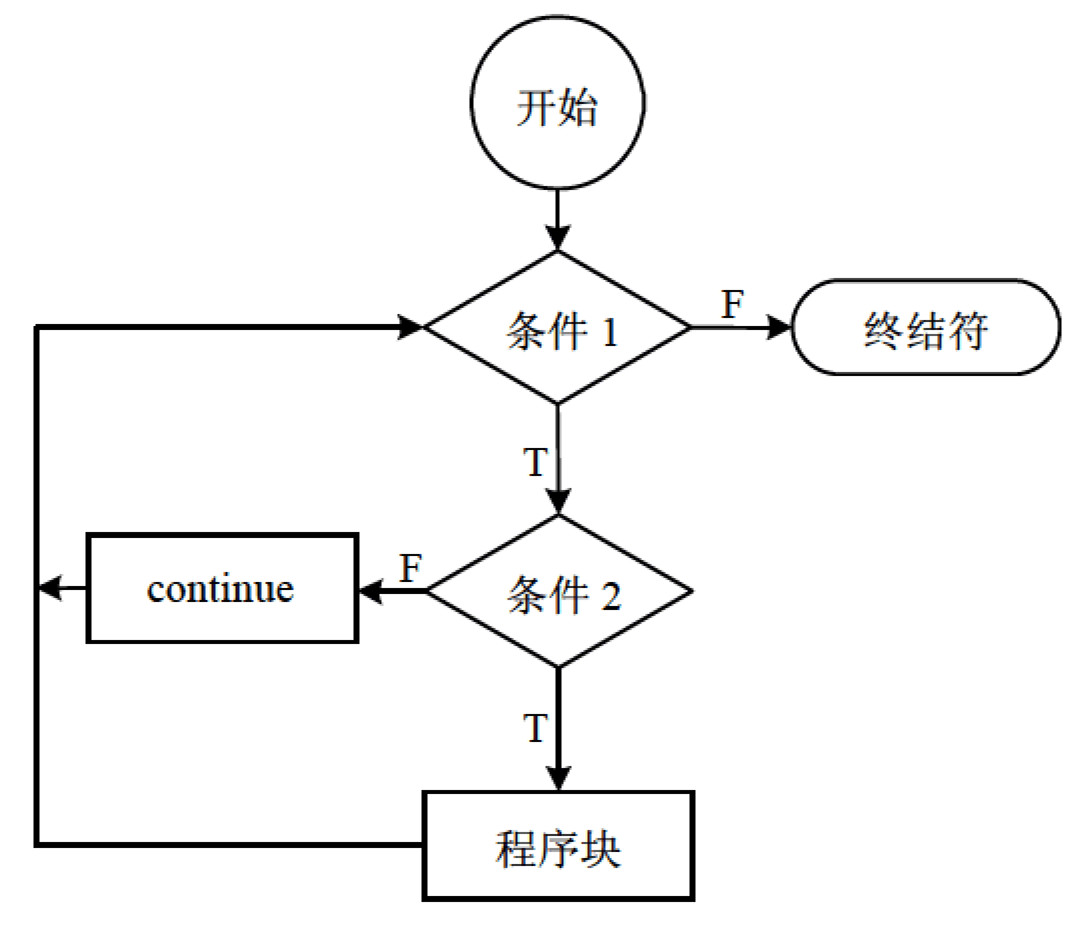


图 4.27　continue 跳转语句的流程图

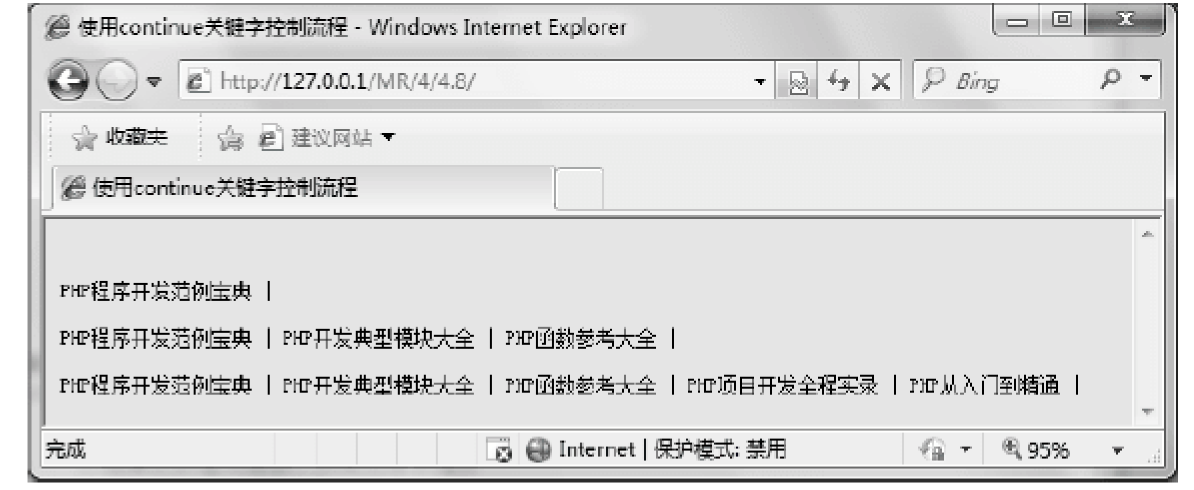
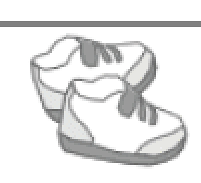
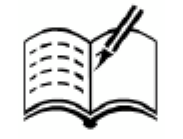


图 4.28　使用 continue 关键字控制流程





指点迷津：

break 和 continue 语句都具有实现跳转的功能，但还是有区别的：continue 语句只是结束本次循环，并不是终止整个循环的执行，而 break 语句则是结束整个循环过程。

多学两招：

什么是流程控制语句的嵌套？

嵌套语句是在流程控制语句中又嵌入控制语句，形象地说就是在大圈中套中圈，中圈中又套了一个小圈。其中嵌套的层次没有限制，如果程序需要可以无限地嵌套下去，只是过多地使用嵌套语句会影响程序运行的速度。

通过嵌套语句可以使程序更加有层次感。嵌套语句的执行顺序是从外到内。嵌套语句可以分为两种类型：一种是同一类型语句中的嵌套，即反复地使用相同的控制语句实现嵌套的功能。另一种是不同的控制语句之间进行嵌套使用，这种方式是最常用的，几乎所有的程序都使用过这种嵌套方法。例如：在 if 条件语句中嵌套 for 循环语句、while 循环语句；在 for 循环语句中嵌套 while 循环语句、foreach 循环语句等，各个流程控制语句之间都可以嵌套使用。通过这种嵌套方法可以使程序更加灵活、实用。

上机演练

上机演练 6　图形计数器

应用 switch 语句和 break 语句设计一个图形计数器，其运行结果如图 4.29 所示。



图 4.29　图形计数器

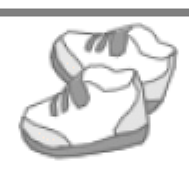
4.5　包含语句

视频讲解：配套资源\mr\4\video\包含语句.exe

引用外部文件可以减少代码的重用性，是 PHP 编程中的重要技巧。PHP 提供了 4 个非常简单却很有用的包含语句，它们允许重新使用任何类型的代码，使用任意一个语句均可将一个文件载入到 PHP 脚本中，从而减少代码的重用性，提高代码维护和更新的效率。

4.5.1　include()语句

使用 include()语句包含外部文件时，只有当代码执行到 include()函数时才将外部文件包含



进来，当所包含的外部文件发生错误时，系统只给出一个警告，而整个 PHP 文件则继续向下执行。其语法如下：

```
void include(string filename);
```

参数 filename 是指定的完整路径文件名。

例 4.9　在设计 Web 页面时，为了保证页面具有一致的外观，可以将网页的头文件和尾文件进行单独存储，然后在其他页面中直接通过 include()语句包含网站的头文件和尾文件即可。如此既减少代码的冗余，也便于对网页头尾文件的维护和更新，其代码如下：（**实例位置：配套资源\mr\4\example\4.9**）

```
<?php include("top.php");?>
<table border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td><img src="images/bg02.gif"></td>
    <td><img src="images/bg03.gif"></td>
  </tr>
</table>
<?php include("bottom.php");?>
```

运行结果如图 4.30 所示。

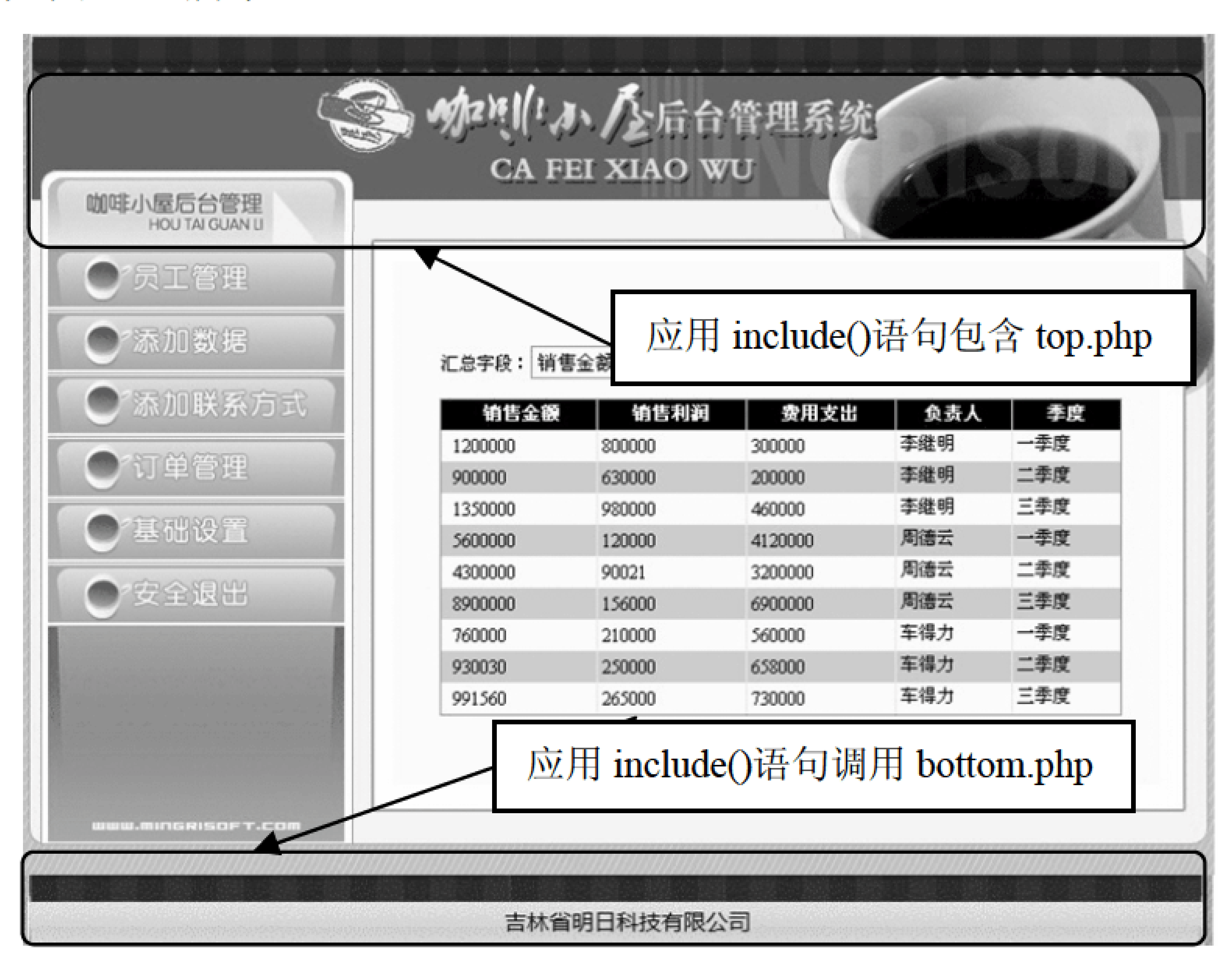


图 4.30　include()语句的应用

4.5.2　require()语句

require()语句与 include()语句类似，都是实现对外部文件的调用。其语法如下：

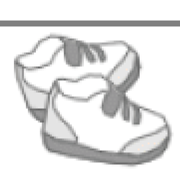
```
void require(string filename);
```

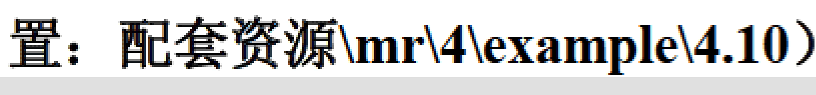
参数 filename 是指定的完整路径文件名。

当使用 require()语句载入文件时，它会作为 PHP 文件的一部分被执行。例如：通过 require()语句载入一个 mr.html 网页文件，那么文件内的任何 PHP 命令都会被处理。但是，如果将 PHP 脚本单纯地放到 HTML 网页中，它是不会被处理的。

通过上面的分析可以看出，PHP 可以使用任何扩展名来命名包含文件，如.inc 文件、html











文件或其他非标准的扩展名文件等。但 PHP 通常用来解析扩展名被定义为.php 文件。建议读者使用标准的文件扩展名。

例 4.10 应用 require()语句包含并运行指定的外部文件 job.php。具体代码如下：（**实例位置：配套资源\mr\4\example\4.10**）

```
<table width="974" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td><?php require("job.php");?></td>                //require语句包含指定文件
  </tr>
</table>
```

job.php 文件的代码如下：

```
<img src="images/top.jpg">
```

运行 index.php 页，输出 require()语句调用的 Web 页面。运行结果如图 4.31 所示。

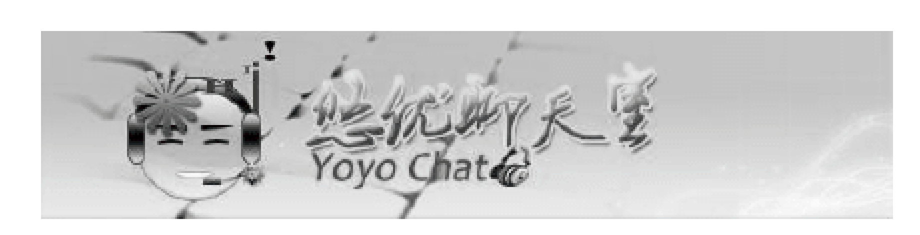


图 4.31　使用 require()语句包含文件

4.5.3　include_once()语句

include_once()语句是 include()语句的延伸，它的作用和 incldue()语句几乎是相同的，唯一的区别在于 include_once()语句会在导入文件前先检测该文件是否在该页面的其他部分被导入过，如果被导入过就不会重复导入该文件，这种区别是非常重要的。例如，要导入的文件中存在一些自定义函数，那么如果在同一个程序中重复导入这个文件，在第二次导入时便会发生错误，因为 PHP 不允许相同名称的函数被重复声明第二次。include_once()函数的语法如下：

```
void include_once (string filename);
```

参数 filename 是指定的完整路径文件名。

例 4.11 应用 include_once()语句包含并运行指定的外部文件 job.php。具体代码如下：（**实例位置：配套资源\mr\4\example\4.11**）

```
<table width="974" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td><?php include_once("job.php");?></td>          //include_once()语句包含指定文件
  </tr>
</table>
```

job.php 文件的代码如下。

```
<img src="images/top.jpg">
```

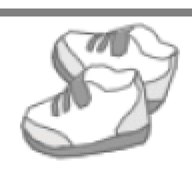
运行结果如图 4.32 所示。



图 4.32　include_once()语句的应用

4.5.4　require_once()语句

require_once()语句是 require()语句的延伸，它的功能与 require()语句基本类似，不同的是，



require_once()语句会先检查要导入的文件是否已经在该程序中的其他地方被调用过，如果被调用过就不会再次重复调用该文件。如果应用 require_once()语句在同一页面中调用两个相同的文件，那么在输出时只有第一个文件被输出，第二次调用的文件不会被输出。require_once()语句的语法如下：

```
void require_once (string filename);
```

参数 filename 是指定的完整路径文件名。

例 4.12　应用 require_once()语句调用外部文件。具体步骤如下：（**实例位置：配套资源\mr\4\example\4.12**）

首先，创建一个动态 PHP 文件，命名为 index.php，然后应用 require_once()函数嵌入 3 个外部文件，代码如下：

```
<table id="__01" width="800" height="632" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td colspan="2">
            <?php require_once("job1.php");?>            //require_once语句包含文件
        </td>
    </tr>
    <tr>
        <td>
            <?php require_once("job2.php");?>
        </td>
        <td>
            <?php require_once("job3.php");?>
        </td>
    </tr>
</table>
```

本范例的运行结果如图 4.33 所示。

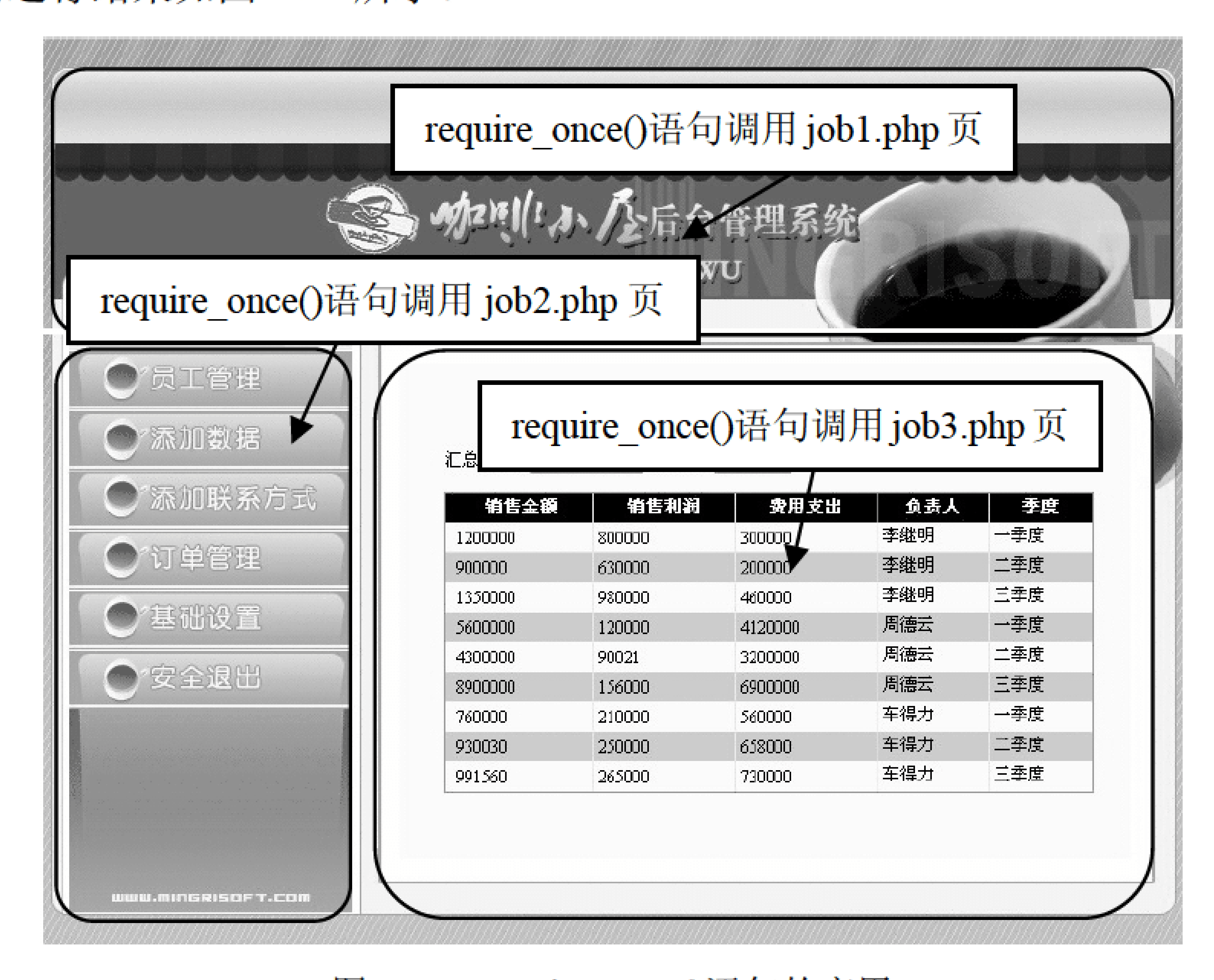
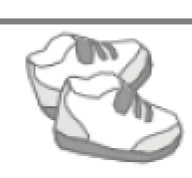
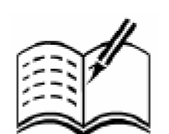
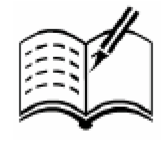


图 4.33　require_once()语句的应用











4.5.5 include()语句和 require()语句的区别

应用 require()语句来调用文件，其应用方法和 include()语句类似，但还存在一定的区别，如下所示。

☑ 在使用 require()语句调用文件时，如果没有找到文件，则 require()语句会输出错误信息，并且立即终止脚本的处理。而 include()语句在没有找到文件时则会输出警告，不会终止脚本的处理。

☑ 使用 require()语句调用文件时，只要程序一执行，会立刻调用外部文件；而通过 incldue()语句调用外部文件时，只有程序执行到该语句时，才会调用外部文件。

例 4.13 为了更好地区分 require()语句和 include()语句的区别，下面通过具体实例进行介绍。**（实例位置：配套资源\mr\4\example\4.13）**

（1）首先建立一个文件 index.php，分别使用 require()语句和 include()语句调用一个不存在的文件 topp.php。

（2）在 index.php 文件中应用 require()语句包含文件，并且指定错误的包含文件“topp.php”。程序代码如下：

```
<?php
echo "明日公司温馨提示：体会require()语句的执行过程";
require("topp.php");                //调用错误的外部文件，以查看require()语句的执行效果
echo "上面的文件名错了，因此弹出错误提示，并且程序终止执行！";
?>
```

脚下留神：

上面是为了看出 require()语句在脚本中的执行效果，将正确的 top.php 文件名称写成错误的 topp.php 文件名。

当运行到 require()语句时，程序调用不存在的 topp.php 文件，因此弹出错误信息，并终止程序的运行，没有输出最后一行字符串。运行结果如图 4.34 所示。

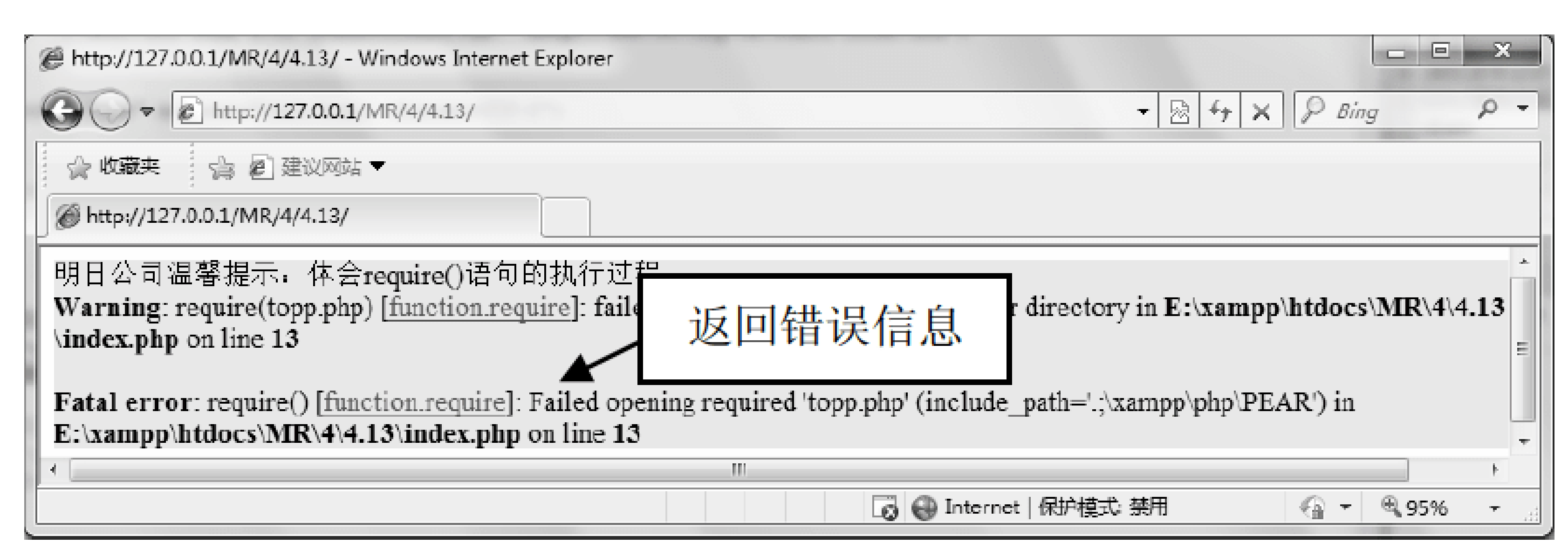


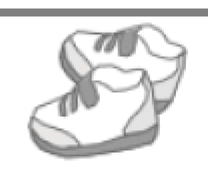
图 4.34 使用 require()语句调用错误的文件

指点迷津：

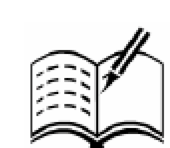
由于 require()语句调用错误的外部文件，因此上面代码段中的“上面的文件名错了，因此弹出错误提示，并且程序终止执行！”字符串未被执行。

（3）在 include.php 文件中应用 include()语句包含外部文件，同样指定错误的文件名称 topp.php。程序代码如下：





```
<?php
echo "明日公司温馨提示：体会include()语句的执行过程";
include("topp.php");                //调用错误的外部文件，以查看include()语句的执行效果
echo "哈哈，虽然上面的文件名错了，但是程序还是会继续执行！我被输出了！";
?>
```



当运行到 include()语句时，由于程序调用不存在的 topp.php 文件，因此弹出警告信息。但与 require()语句不同的是，该程序没有终止执行，继续处理脚本文件，直到输出最后一行字符串为止，运行结果如图 4.35 所示。

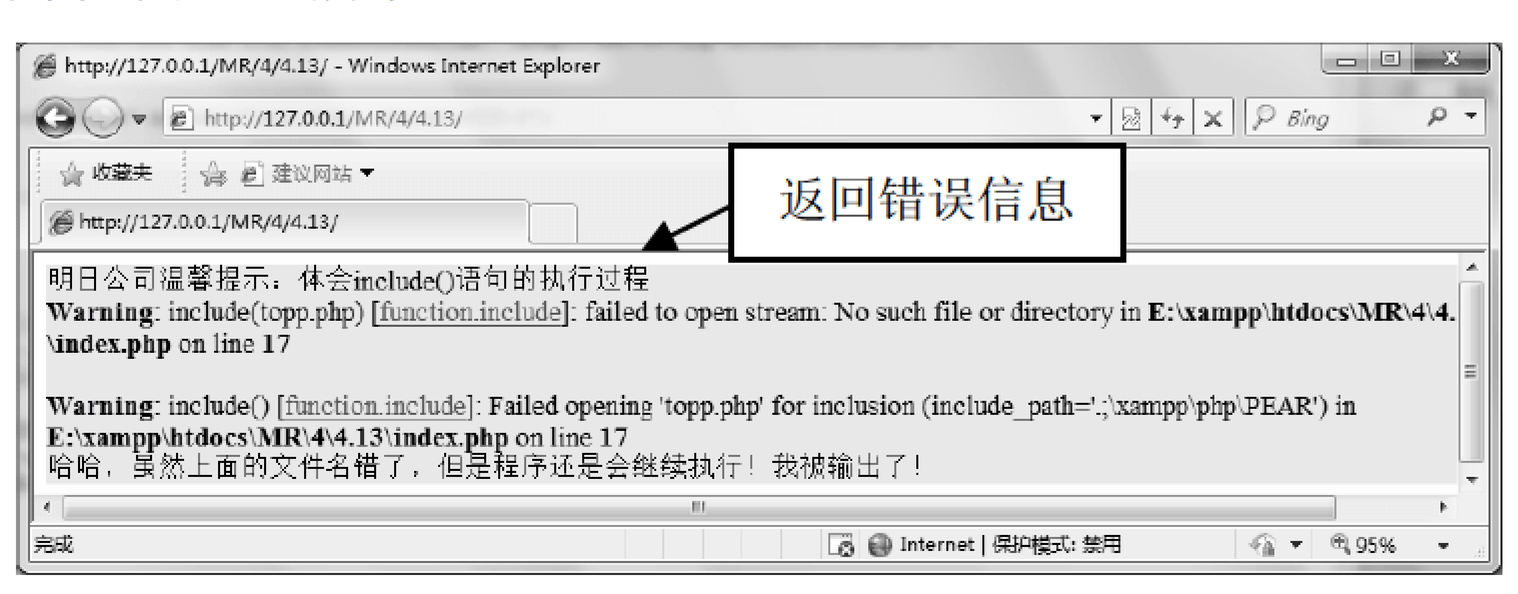


图 4.35　使用 include()语句调用错误的文件

上机演练

上机演练 7　包含数据库连接文件

一个程序可能要与数据库多次交互，所以数据库连接信息要单独保存在一个文件中，这也是代码重用的一种体现。通过 include()语句包含数据库文件，运行结果如图 4.36 所示。

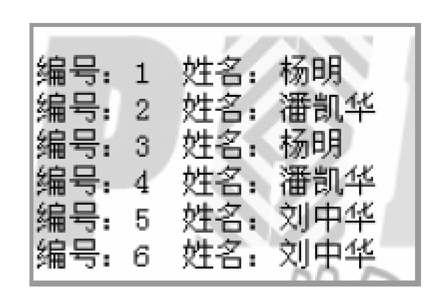


图 4.36　包含数据库连接文件

本 章 摘 要

1．if、switch 条件控制语句的语法讲解以及实际的应用。

2．while、do…while、for、foreach 语句循环控制语句的语法、特点以及它们之间的区别，同时还以示例、上机演练和实战模拟等形式介绍它们的实际应用。

3．break、continue 语句的应用。

4．包含语句 include()、require()等的应用。

习　　题

1．以下代码的运行结果为（　　）。

```
<?php
    if($i=""){
```




```
            echo "a";
        }else{
            echo "b";
        }
?>
```

A．输出 a　　B．输出 b

C．条件不足，无法确定　　D．运行出错

2．以下哪一个语言结构最能代表以下多路分支 if 语句的选择结构？（　　）

```
<?php
    if($a=='a'){
        somefunction();
        }else if($a=='b'){
            anotherfunction();
        }else if($a=='c'){
            dosomething();
        }else{
            donothing();
    }
?>
```

A．没有 default 的 switch 语句　　B．一个递归的函数调用

C．一个 while 语句　　D．使用 default 的 switch 语句

3．以下代码返回的结果为（　　）。

```
<?php
    function p(){
        return 1;
    }
    if(p()){
        echo"fasle";
        }else{
            echo"true";
        }
?>
```

A．true　　B．false

C．程序运行出错　　D．根据版本来定

4．以下程序运行结果为（　　）。

```
<?php
    $var=FALSE;
    if(empty($var)){
        echo"null";
    }else{
        echo"have value";
    }
?>
```

A．null　　B．have value

C．无法确定　　D．什么也不显示，提示错误



5．指出下面的代码是否有问题，如果有问题，错误有几处（　　）。

①if test num1 –gt num2

②then

③echo “Test”

④elseif

⑤echo “test”

⑥fi

A．正确，该代码没有问题　　　B．有两处错误

C．有 3 处错误　　　　　　　　D．有 4 处错误

6．include()和 require()语句都能把另外一个文件包含到当前文件中，它们的区别是（　　）；为了避免多次包含同一文件，可以用语句（　　）来代替它们。

7．程序设计的结构大致可以分为（　　）种，它们分别是（　　）。

8．下面代码的运行结果为（　　）。

```
<?php
    $sum=0;
    for ($i=1;$i<=10;$i++){
    if($i>5)break;
    $sum=$sum+$i;
    }
    echo"sum=".$sum;
?>
```

9．以下程序的运行结果为（　　）。

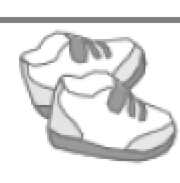
```
<?php
    $op1=1;
    $sum=0;
    do{
    $sum=$sum+$op1;
    $op1=$op1+1;
    }while($op1<=10);
    echo"sum=".$sum;
?>
```

10．以下程序运行结果为（　　）。

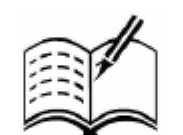
```
<?php
    $num=1;
    while($num!=1){
        echo"不会看到.";
    }
    do{
        echo"会看到。";
    }while($num!=1);
?>
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此



来检验大家对本章知识的掌握情况，同时给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 3 章内容，其中对所列实例的实现方法进行了详细讲解）。



Note

实战模拟 1　健康生活提醒

一些手机在开机时会出现开机问候语或者提示今日日程。通过 switch()语句根据当前日期，给出健康生活提示信息，运行结果如图 4.37 所示。

实战模拟 2　if 语句判断美女征婚条件

网络上有一个美女征婚的笑话，其主要内容为：一女子让电脑为其征婚，开出的征婚条件有两点。

（1）要帅。

（2）要有车。

电脑为她搜寻结果：象棋。

这位女子不愿接受此次搜寻的结果，重新输入条件。

（1）要有很多钱。

（2）要有漂亮的房子。

电脑为她再次搜寻结果：银行。

女子无语，于是再次输入条件。

（1）要有安全感。

（2）要长得酷。

电脑为她搜寻结果：奥特曼。

女子崩溃，但是仍不死心，于是最后一次输入条件。

（1）要有很多钱，要长得酷，要有安全感。

（2）有车，要帅，要有漂亮的房子。

过了许久，电脑为她搜寻结果：奥特曼在银行里下象棋。

将这个美女征婚的笑话改编成一个小程序，应用 if 条件语句对美女提交的问题进行判断，并且给出答案。其运行结果如图 4.38 所示。

今天是黑色星期五。。。。。。

图 4.37　健康生活提醒

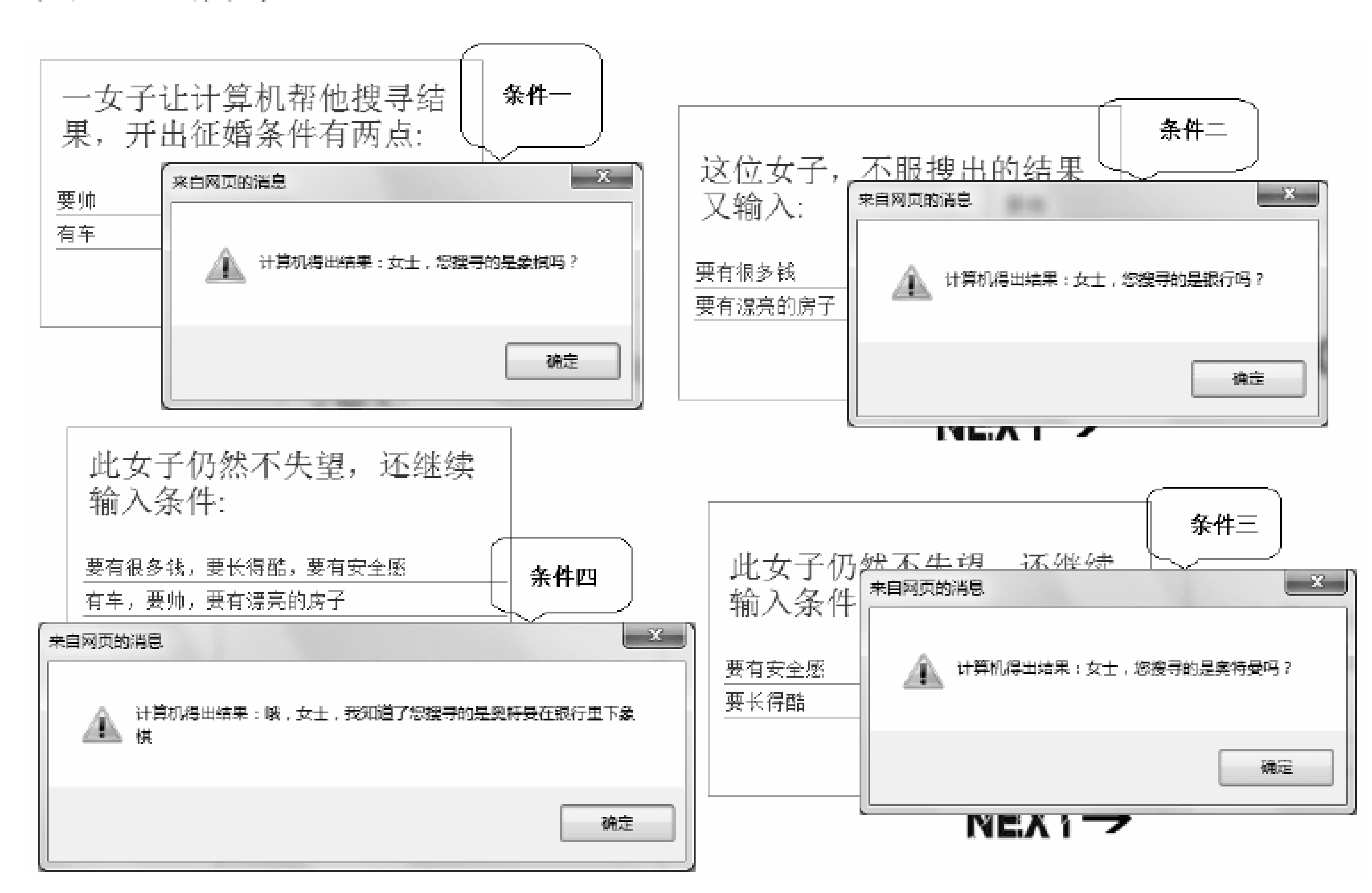


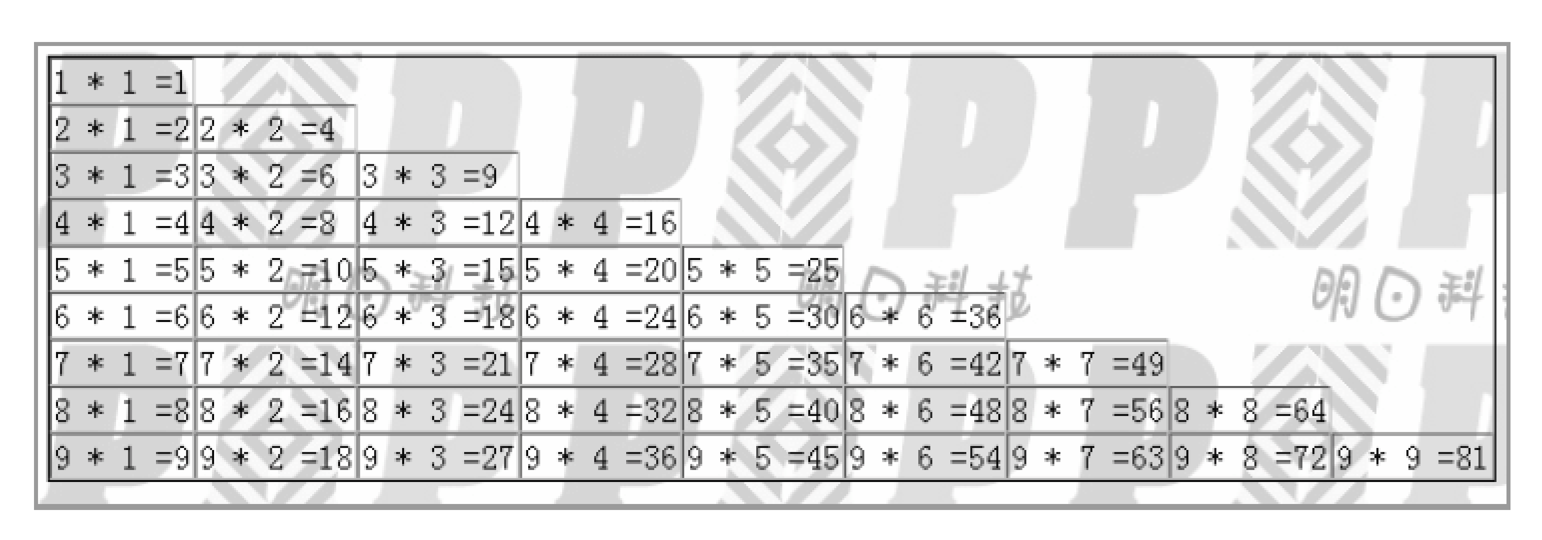
图 4.38　if 语句判断美女征婚条件





实战模拟 3　网页版九九乘法表

在编程世界里算法是一门独立于语法、函数之外的知识，它着重考验用户的逻辑思维能力并且与数学知识息息相关。通过 for 循环语句创建一个九九乘法表，运行结果如图 4.39 所示。



1 * 1 =1								
2 * 1 =2	2 * 2 =4							
3 * 1 =3	3 * 2 =6	3 * 3 =9						
4 * 1 =4	4 * 2 =8	4 * 3 =12	4 * 4 =16					
5 * 1 =5	5 * 2 =10	5 * 3 =15	5 * 4 =20	5 * 5 =25				
6 * 1 =6	6 * 2 =12	6 * 3 =18	6 * 4 =24	6 * 5 =30	6 * 6 =36			
7 * 1 =7	7 * 2 =14	7 * 3 =21	7 * 4 =28	7 * 5 =35	7 * 6 =42	7 * 7 =49		
8 * 1 =8	8 * 2 =16	8 * 3 =24	8 * 4 =32	8 * 5 =40	8 * 6 =48	8 * 7 =56	8 * 8 =64	
9 * 1 =9	9 * 2 =18	9 * 3 =27	9 * 4 =36	9 * 5 =45	9 * 6 =54	9 * 7 =63	9 * 8 =72	9 * 9 =81

图 4.39　网页版九九乘法表

实战模拟 4　读取购物车中的数据

用户在进行开发时，小型的数据没有必要与数据库进行交互，可以直接将数据存入数组，这样不仅可以节约开发时间，而且还可以节省服务器资源。通过对数组函数的相关操作实现读取数组购物车中的数据，运行结果如图 4.40 所示。



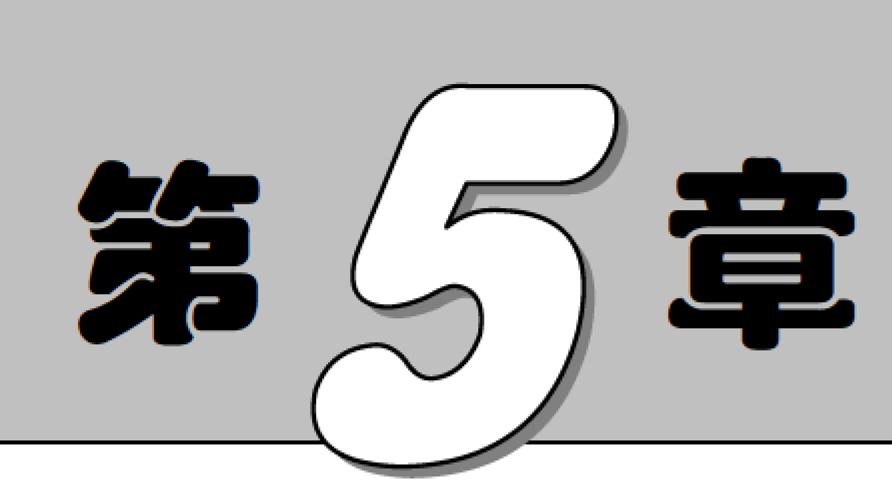
图 4.40　读取购物车中的数据

实战模拟 5　多图片上传

上传图片是很多娱乐网站所必须包含的模块，其优点是与用户有良好的互动，由用户选择喜欢的图片或文件上传。使用 for 循环语句实现多图片上传功能，一次可以最多上传 4 张图片。运行结果如图 4.41 所示。



图 4.41　多图片上传



PHP 数组

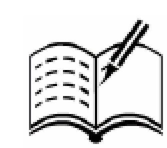
（自学视频、源程序：配套资源\mr\5\）

数组提供了一种快速、方便地管理一组相关数据的方法，是 PHP 程序设计中的重要内容。通过数组可以对大量性质相同的数据进行存储、排序、插入及删除等操作，从而可以有效地提高程序开发效率及改善程序的编写方式。在本章中将对 PHP 中的数组操作技术进行系统、详细地讲解。

学习摘要：

- 数组概述
- 数组类型
- 声明数组
- 遍历、输出数组
- 统计数组元素个数
- 向数组中添加元素
- 获取数组中最后一个元素
- 删除重复数组
- 获取数组中指定元素的键值
- 将数组中元素合成字符串





5.1 数组概述

数组是一组数据的集合，将数据按照一定规则组织起来，形成一个可操作的整体，是对大量数据进行有效组织和管理的手段之一。通过数组函数可以对大量性质相同的数据进行存储、排序、插入及删除等操作，从而可以有效地提高程序开发效率及改善程序的编写方式。

数组的本质是储存、管理和操作一组变量。数组与变量的比较效果如图 5.1 所示。

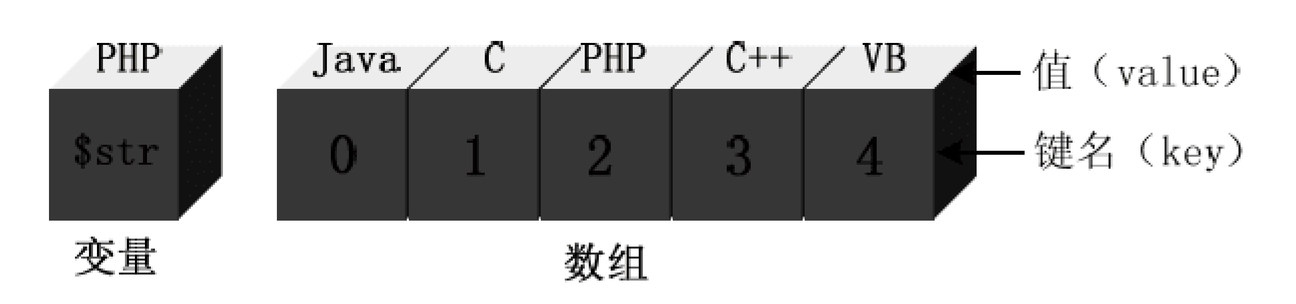


图 5.1 数组与变量的比较效果

5.2 数组类型

在 PHP 中将数组分为一维数组、二维数组和多维数组，但是无论是一维数组还是多维数组，都可以分为数字索引数组（indexed array）和关联数组（associative array）两种类型。数字索引数组使用数字作为键名（如图 5.1 所示就是一个数字索引数组），关联数组使用字符串作为键名，如图 5.2 所示。

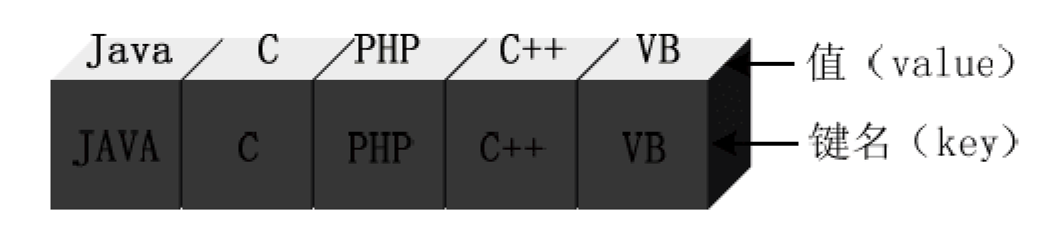


图 5.2 关联数组

（1）数字索引数组

数字索引数组，下标（键名）由数字组成，默认从 0 开始，每个数字对应数组元素在数组中的位置，不需要特别指定，PHP 会自动为数字索引数组的键名赋一个整数值，然后从这个值开始自动增量。当然，也可以指定从某个具体位置开始保存数据。

数组中的每个实体都包含两项：键名和值。可以通过键名来获取相应数组元素（值），如果键名是数值那么就是数字索引数组；如果键名是数值与字符串的混合，那么就是关联数组。

下面创建一个数字索引数组，代码如下：

```
$arr_int = array ("PHP入门与实战","C#入门与实战","VB入门与实战");    //声明数字索引数组
```

（2）关联数组

关联数组，下标（键名）由数值和字符串混合的形式组成。如果在一个数组中，有一个键名不是数字，那么这个数组就称为关联数组。

关联数组使用字符串键名来访问存储在数组中的值，如图 5.2 所示。

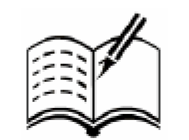
下面创建一个关联索引数组，代码如下：

```
$arr_string = array ("PHP"=>"PHP入门与实战","JAVA"=>"JAVA入门与实战","C#"=>"C#入门与
实战");   //声明关联数组
```



指点迷津：

关联数组的键名可以是任何一个整数或字符串。如果键名是一个字符串，则要给这个键名或索引加上个定界修饰符——单引号（'）或双引号（"）。对于数字索引数组，为了避免不必要的麻烦，最好也加上定界符。



Note

5.3 声明数组

视频讲解：配套资源\mr\5\video\声明数组.exe

在讲解数组的声明方法之前，先来了解一下数组的命名规则。PHP 中声明数组的规则如下：

（1）数组的名称由一个美元符号开始，第一个字符是字母或下划线，其后是任意数量的字母、数字或下划线。

（2）在同一个程序中，标量变量和数组变量都不能重名。例如：如果已经存在一个名称为$string 的变量，而又创建一个名称为$string 的数组，那么前一个变量就会被覆盖。

（3）数组的名称区分大小写。例如：$String 与$string 是不同的。

声明数组的方法有两种，分别为用户声明和函数声明。下面将具体介绍。

5.3.1 用户创建数组

用户创建数组应用的是标识符[]，通过标识符[]可以直接为数组元素赋值。其基本格式如下：

```
$arr['key'] = value;
$arr['0'] = value;
```

其中 key 可以是 int 型或者字符串型数据，value 可以是任何值。

例 5.1 应用标识符[]创建数组 array，然后应用 print_r()函数输出数组元素，代码如下：**（实例位置：配套资源\mr\5\example\5.1）**

```
<?php
$array['0']="PHP入门与实战";          //通过标识符[]定义数组元素值
$array['1']="JAVA入门与实战";         //通过标识符[]定义数组元素值
$array['2']="VB入门与实战";           //通过标识符[]定义数组元素值
$array['3']="VC入门与实战";           //通过标识符[]定义数组元素值
print_r($array);                     //输出所创建数组的结构
?>
```

运行结果如图 5.3 所示。

多学两招：

本例中使用 print_r()函数输出数组元素，因为使用 print_r 输出数组，将会按照一定格式输出数组中所有的键名和元素。而使用 echo 语句可以输出数组中指定的某个元素，下面使用 echo 语句输出数组中的第一个元素，如图 5.4 所示。

c创建数组

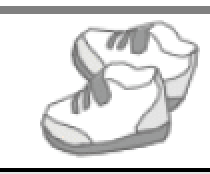
Array ([0] => PHP入门与实战 [1] => JAVA入门与实战 [2] => VB入门与实战 [3] => VC入门与实战)

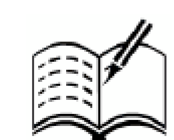
图 5.3 通过标识符[]创建的数组结构

c创建数组

PHP入门与实战

图 5.4 使用 echo 语句输出数组





指点迷津：

（1）用户创建数组比较适合创建不知大小或者大小可能发生改变的数组。（2）切忌在通过标识符[]直接为数组元素赋值，同一数组元素中的数组名称必须相同。

5.3.2　函数创建数组

PHP 中最常用的创建数组的函数是 array()，其语法如下：

```
array array ( [mixed ...])
```

参数 mixed 的格式为“key => value”，多个参数 mixed 用逗号分隔，分别定义键名（key）和值（value）。

应用 array()函数声明数组时，数组下标（键名）既可以是数值索引也可以是关联索引。下标与数组元素值之间用“=>”进行连接，不同数组元素之间用逗号进行分隔。

应用 array()函数定义数组时，可以在函数体中只给出数组元素值，而不必给出键名。

指点迷津：

（1）数组中的索引（key）可以是字符串或数字。如果省略了索引，会自动产生从 0 开始的整数索引。如果索引是整数，则下一个产生的索引将是目前最大的整数索引+1。如果定义了两个完全相同的索引，则后面一个会覆盖前一个。（2）数组中的各数据元素的数据类型可以不同，也可以是数组类型。当 mixed 是数组类型时，就是二维数组。

例 5.2　应用 array()函数声明数组，并输出数组中的元素，代码如下：**（实例位置：配套资源\mr\5\example\5.2）**

```
<?php
$arr_string=array('one'=>'php','two'=>'java');      //以字符串作为数组索引，指定关键字
print_r($arr_string);                               //通过print_r()函数输出数组
echo "<br>";
echo $arr_string['one']."<br>";                     //输出数组中的索引为java的元素
$arr_int=array('php','java');                       //以数字作为数组索引，从0开始，没有指定关键字
print_r($arr_int);                                  //输出整个数组
echo "<br>";
echo $arr_int['0']."<br>";                          //输出数组中的第1个元素
$arr_key=array(0 =>'PHP入门与实战', 1 =>'JAVA入门与实战', 1 =>'VB入门与实战');//指定相同的索引
print_r($arr_key);                                  //输出整个数组，发现只有两个元素
?>
```

运行结果如图 5.5 所示。

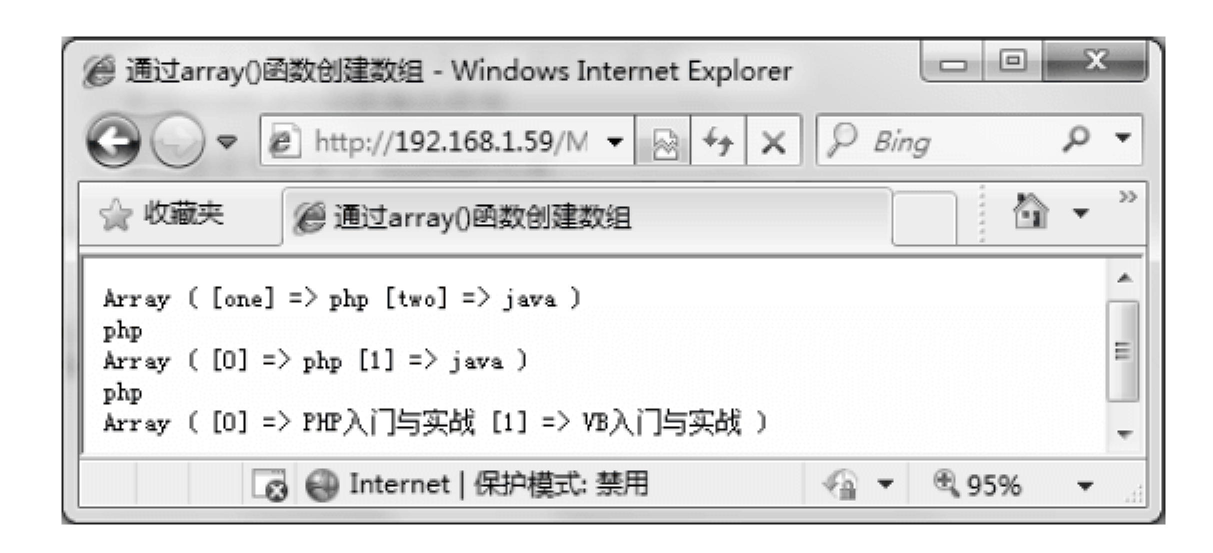
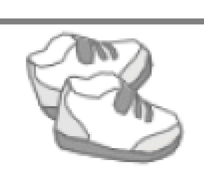
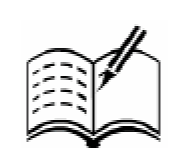


图 5.5　查看数组的结构



5.3.3　创建二维数组



Note

上述创建的数组都是只有一列数据内容的，因此称为一维数组。如果将两个一维数组组合成一个数组，那么就称为二维数组。

例 5.3　应用 array()函数创建一个二维数组，并输出数组的结构，代码如下：（**实例位置：配套资源\mr\5\example\5.3**）

```
<?php
$str = array (
        "网络编程图书"=>array ("PHP入门与实战","C#入门与实战","VB入门与实战"),
        "历史图书"=>array ("1"=>"春秋","2"=>"战国","3"=>"左传"),
        "文学图书"=>array ("明朝那些事儿",3=>"狼图腾","鬼吹灯")
 );                    //声明二维数组
print_r ($str) ;       //输出数组元素
?>
```

运行本实例，查看运行结果的源文件如图 5.6 所示。

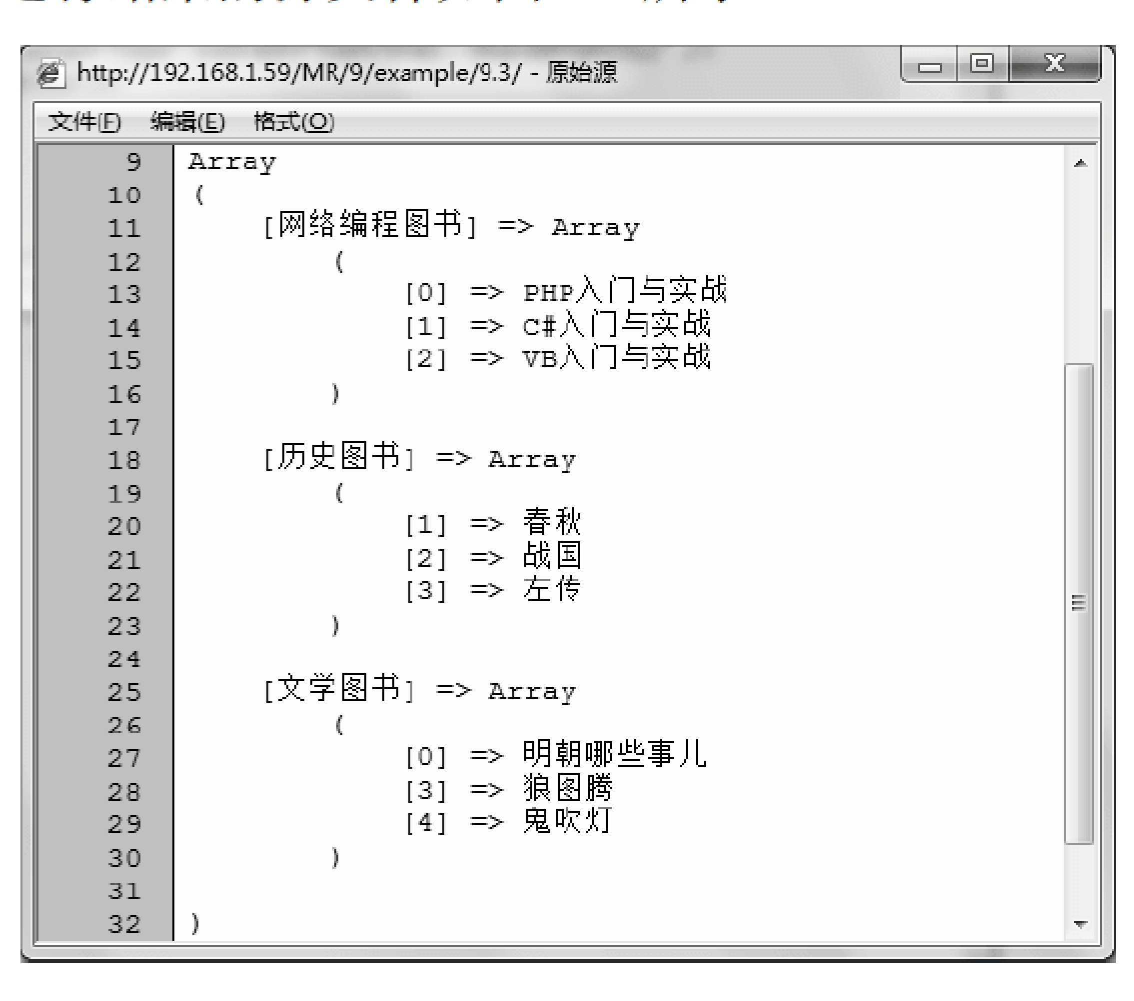


图 5.6　二维数组的结构

多学两招：

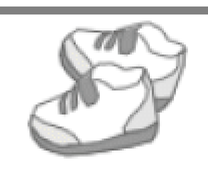
根据创建二维数组的方法，举一反三，就可以很容易地创建三维、四维或多维数组。

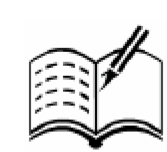
5.4　遍历、输出数组

视频讲解：配套资源\mr\5\video\遍历数组.exe

5.4.1　遍历数组

遍历数组就是按照一定的顺序依次访问数组中的每个元素，直到访问完为止。PHP 中可以通过流程语句（foreach 和 for 循环语句）和函数（list()和 each()）来遍历数组，下面分别介绍这几种遍历数组的方法。





1. foreach 语句遍历数组

在第 4 章中已经介绍了 foreach 语句的循环结构，下面使用该语句来遍历数组。

例 5.4 使用 foreach 语句遍历一维数组$str，具体代码如下：（**实例位置：配套资源\mr\5\example\5.4**）

```
<?php
//创建数组
$str=array('编程词典网'=>'www.mrbccd.com',
           '明日图书网'=>'www.mingribook.com',
);
echo "原数组：";
print_r($str);
echo "<br>";
echo "遍历后的值：";
foreach($str as $link){          //遍历数组
    echo $link."  ";
}
?>
```

运行结果如图 5.7 所示。



图 5.7 使用 foreach 语句遍历数组（遍历值）

例 5.4 中是将数组的值遍历输出，下面将数组的键名和元素值都遍历输出，只需将例 5.4 中的 foreach 循环语句改为：

```
foreach($str as $key=>$link){
    echo "$key----$link"."<br>";        //对应输出数组中的键名和元素值
}
```

运行结果如图 5.8 所示。

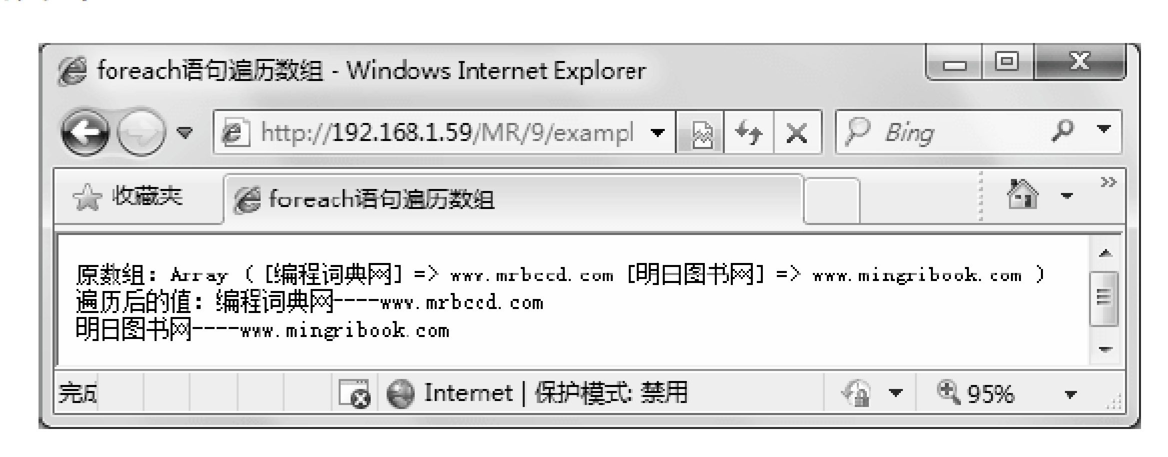
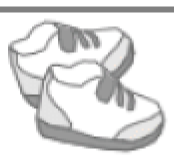
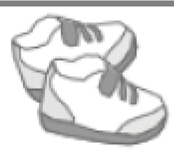


图 5.8 使用 foreach 语句遍历数组（遍历键名和值）

例 5.5 如果想要遍历多维数组，只要使用两个 foreach 语句就可以了，具体代码如下：（**实例位置：配套资源\mr\5\example\5.5**）

```
<?php
$str = array (
    "网络编程图书"=>array ("PHP入门与实战","C#入门与实战","VB入门与实战"),
```

```
            "历史图书"=>array ("1"=>"春秋","2"=>"战国","3"=>"左传"),
            "文学图书"=>array ("明朝那些事儿",3=>"狼图腾","鬼吹灯")
        );                              //二维数组
foreach($str as $key=>$link){           //使用foreach语句获取一维数组的元素和值
    foreach($link as $value){ //使用foreach语句获取上一个foreach语句遍历得到的元素和值中的值
        echo $value."<br>";
    }
}
?>
```

运行结果如图 5.9 所示。

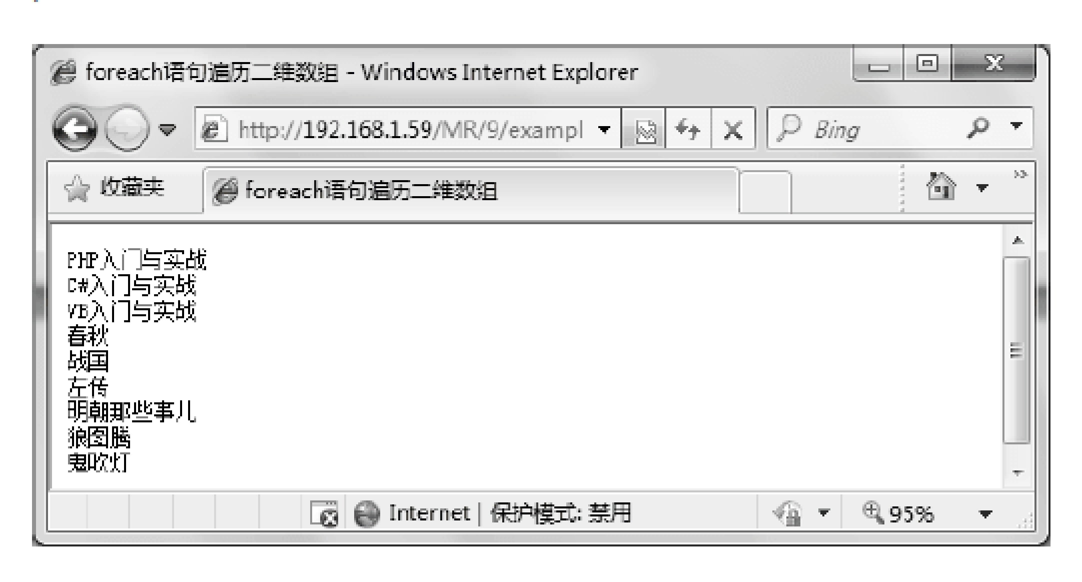


图 5.9　使用 foreach 遍历多维数组

2. for 语句遍历数组

如果要遍历的数组是数字索引数组，并且数组的索引值为连续的整数时，可以使用 for 循环来遍历，但前提条件是需要应用 count()函数获取到数组中元素的数量，然后将获取的元素数量作为 for 循环执行的条件，才能完成数组的遍历。

例 5.6　使用 for 循环来遍历数组$array，代码如下：（**实例位置：配套资源\mr\5\example\5.6**）

```
<?php
$array=array(                           //定义数组
        "0"=>"PHP入门与实战",
        "1"=>"JAVA入门与实战",
        "2"=>"VB入门与实战",
        "3"=>"VC入门与实战"
        );
for($i=0;$i<count($array);$i++){        //使用for循环遍历数组
    echo $array[$i]."<br>";             //输出数组元素
}
?>
```

运行结果如图 5.10 所示。

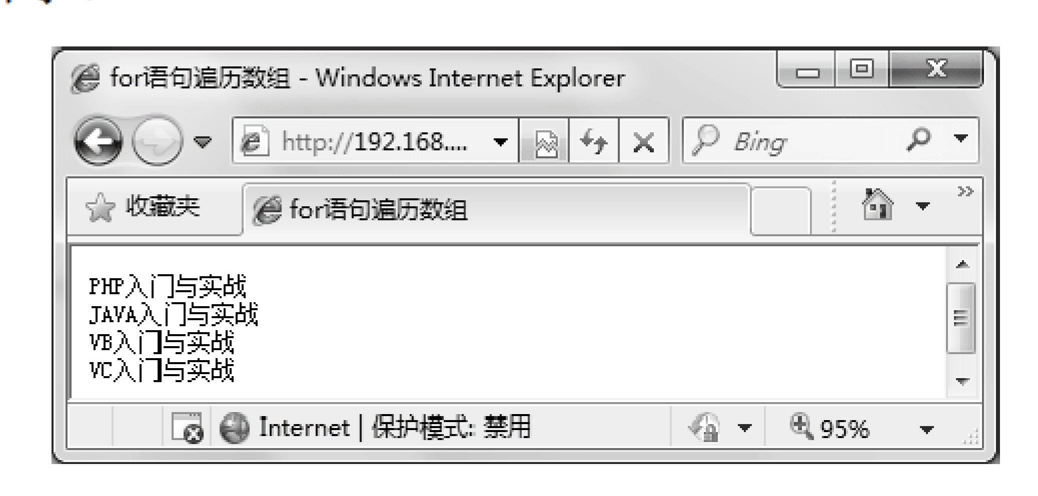
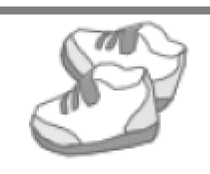
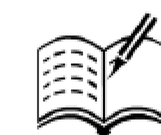


图 5.10　使用 for 循环遍历数组





3. 通过数组函数 list()和 each()遍历数组

☑ list()函数

list()函数将数组中的值赋给一些变量，该函数仅能用于数字索引的数组，且数字索引从 0 开始。语法如下：

```
void list ( mixed ...)
```

参数 mixed 为被赋值的变量名称。

☑ each()函数

each()函数返回数组中的键名和对应的值，并向前移动数组指针。其语法如下：

```
array each ( array array)
```

参数 array 为输入的数组。

例 5.7　使用 list()和 each()函数来遍历数组$array，具体代码如下：（**实例位置：配套资源\mr\5\example\5.7**）

```
<?php
$array=array(                                //定义数组
        "0"=>"PHP入门与实战",
        "1"=>"JAVA入门与实战",
        "2"=>"VB入门与实战",
        "3"=>"VC入门与实战"
        );
/*
使用list函数获取each函数中返回数组的值
并分别赋给$name和$value，然后使用while循环输出
*/
while(list($name,$value)=each($array)){
    echo $name=$value."<br>";          //输出list函数获取到的键名和值
}
?>
```

运行结果如图 5.11 所示。

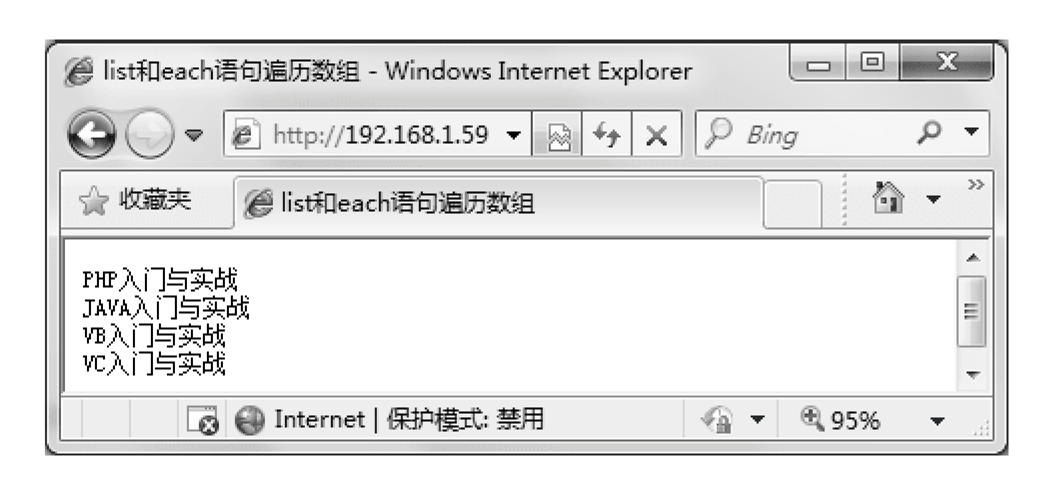


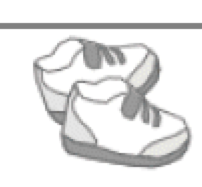
图 5.11　通过数组函数 list()和 each()遍历数组

5.4.2　输出数组元素

在前面已经实践过数组的输出，即 print_r()函数和 echo 语句。

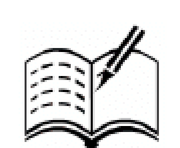
☑ print_r()函数可以输出数组的结构，也可以使用 var_dump()函数，同样是输出数组的结构。

☑ echo 语句则是单纯地输出数组中的某个元素，而且要有标识符[]和数组索引的配合，其格式是“echo $array[0]”。同样还有 print 语句，它也可以单纯地输出数组中的某



个元素值。

5.5 PHP 数组函数



Note

视频讲解：配套资源\mr\5\video\PHP 数组函数.exe

下面介绍一些在实际程序开发中比较常用的数组函数。如果读者需要使用数组函数实现某些特殊的功能，建议参考 PHP 中文手册，那里有对所有数组函数的详细介绍，可以根据所要实现的功能进行选择、学习或研究。

5.5.1 统计数组元素个数

在 PHP 中，应用 count()函数可以对数组中的元素个数进行统计，在讲解使用 for 循环遍历数组时已经应用到。下面对该函数进行详细介绍，语法格式如下：

```
int count ( mixed var [, int mode])
```

参数 var 指定操作的数组对象；参数 mode 为可选参数，如果 mode 的值设置为 COUNT_RECURSIVE（或 1），默认值为 0；count()函数检测多维数组，该函数返回数组元素的个数。

多学两招：

如果 count()函数的操作对象是“NULL”，那么返回结果是 0。count()函数对没有初始化的变量返回 0，但对于空的数组也会返回 0。如果要判断变量是否初始化，则可以应用 isset()函数。count()函数不能识别无限递归。

例 5.8 使用 count()函数统计数组中元素个数，并输出统计结果。代码如下：**（实例位置：配套资源\mr\5\example\5.8）**

```
<?php
$array=array(0 =>'PHP入门与实战', 1 =>'JAVA入门与实战', 2 =>'VB入门与实战',3=>"VC入门与实战");
echo count($array);             //统计数组中元素个数，并使用echo语句输出统计结果
?>
```

运行结果为：4

5.5.2 向数组中添加元素

在 PHP 中，使用 array_push()函数可以向数组中添加元素，将传入的元素添加到某个数组的末尾，并返回数组新的单元总数。语法如下：

```
int array_push ( array array, mixed var [, mixed ...])
```

参数 array 为指定的数组；参数 var 为压入数组中的值。

例 5.9 使用 array_push()函数向数组中添加元素，并输出添加元素后的数组。代码如下：**（实例位置：配套资源\mr\5\example\5.9）**

```
<?php
$array=array(0 =>'PHP入门与实战', 1 =>'JAVA入门与实战');   //声明数组
echo "添加前的数组元素：";
```



```
print_r($array);
echo "<br>";
array_push($array,'VB入门与实战','VC入门与实战');          //向数组中添加元素
echo "添加后的数组元素：";
print_r($array);                                          //输出添加后的数组结构
?>
```



运行结果如图 5.12 所示。



图 5.12　使用 array_push()函数向数组中添加元素

5.5.3　获取数组中最后一个元素

在 PHP 中，通过 array_pop()函数可以获取并返回数组中的最后一个元素，并将数组的长度减一，如果数组为空（或者不是数组）将返回 null。语法如下：

```
mixed array_pop ( array array)
```

参数 array 为输入的数组。

例 5.10　首先应用 array_push()函数向数组中添加元素，然后应用 array_pop()函数获取数组中最后一个元素，最后输出最后一个元素值。代码如下：（**实例位置：配套资源\mr\5\example\5.10**）

```
<?php
$array=array(0 =>'PHP入门与实战', 1 =>'JAVA入门与实战');   //声明数组
array_push($array,'VB入门与实战','VC入门与实战');          //向数组中添加元素
$last_array=array_pop($array);                            //获取数组中最后一个元素
echo $last_array;                                         //返回结果为VC入门与实战
?>
```

运行结果为：VC 入门与实战

5.5.4　删除数组中重复元素

在 PHP 中，使用 array_unique()函数可以将数组中重复的元素删除，语法如下：

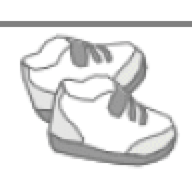
```
array array_unique ( array array)
```

参数 array 为输入的数组。

脚下留神：

虽然 array_unique()函数只保留重复值的第一个键名。但是，这第一个键名并不是在未排序的数组中同一个值的第一个出现的键名，只有当两个字符串的表达式完全相同时（(string) $elem1 === (string) $elem2），第一个单元才被保留。

例 5.11　首先定义一个数组，然后应用 array_push()函数向数组中添加元素，并输出数组，最后应用 array_unique()函数，删除数组中重复元素，并输出数组。代码如下：（**实例位置：配**



套资源\mr\5\example\5.11）**

```
<?php
$arr_int = array ("PHP", "JAVA","VC");          //定义数组
array_push ($arr_int, "PHP","VC");              //向数组中添加元素
print_r($arr_int);                              //输出添加后的数组
$result=array_unique($arr_int);                 //删除添加后数组中重复的元素
print_r($result);                               //输出删除重复元素后的数组
?>
```

运行结果如图 5.13 所示。



图 5.13　删除数组中重复元素

多学两招：

使用 unset()函数可删除数组中的某个元素，例如将例 5.11 中$arr_int 数组的第 2 个元素删除，代码如下：

```
unset($arr_int[1]);
```

5.5.5　获取数组中指定元素的键名

获取数组中指定元素的键名可以使用 array_search()函数或者 array_keys()函数。

（1）array_search()函数可获取数组中指定元素的键名。成功返回元素的键名，否则返回 false。其语法如下：

```
mixed array_search ( mixed needle, array haystack [, bool strict])
```

array_search()函数的参数说明如表 5.1 所示。

表 5.1　array_search()函数的参数说明

参　　数	说　　明
needle	指定在数组中搜索的值，如果 needle 是字符串，则以区分大小写的方式进行
haystack	指定被搜索的数组
strict	可选参数，如果值为 True，则还将在 haystack 中检查 needle 的类型

指点迷津：

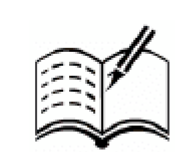
array_search()函数是区分大小写的。

例 5.12　使用 array_search()函数获取数组中元素的键名，具体代码如下：（**实例位置：配套资源\mr\5\example\5.12）**

```
<?php
$arr=array("苹果","桔子","香蕉","梨"); //创建数组，数组中有4个元素
```



```
        $name=array_search("香蕉",$arr);        //使用array_search获取$arr数组中“香蕉”的键名，然后将
获取的结果赋给$name变量
        echo $name;                             //输出结果
        ?>
```



运行结果为：2

（2）array_keys()函数获取数组中重复元素的所有键名。如果查询的元素在数组中出现两次以上，那么 array_search()函数则返回第一个匹配的键名。如果想要返回所有匹配的键名，则需要使用 array_keys()函数。语法如下：

```
array array_keys ( array input [, mixed search_value [, bool strict]] )
```

array_keys()返回 input 数组中的数字或字符串的键名。如果指定可选参数 search_value，则只返回该值的键名；否则 input 数组中的所有键名都会被返回。

例 5.13　使用 array_keys()函数来获取数组中重复元素的所有键名，具体代码如下：（**实例位置：配套资源\mr\5\example\5.13**）

```
<?php
$arr=array("苹果","桔子","香蕉","梨","香蕉");
$name=array_keys($arr,"香蕉");          //使用array_keys获取$arr数组中“香蕉”的所有键值
print_r($name);                 //因为array_keys函数返回的是数组类型的值，所以使用print_r输出
?>
```

运行结果为：array ([0] => 2 [1] => 4)

5.5.6　将数组中元素合成字符串

通过字符串函数 explode()可以将字符串分割成数组，而通过数组函数 implode()可以将数组中的元素组合成一个新字符串。implode()函数的语法如下：

```
string implode(string glue, array pieces)
```

参数 glue 是字符串类型，指定分隔符；参数 pieces 是数组类型，指定要被合并的数组。

例如：应用 implode()函数将数组中的内容以*为分隔符进行连接，从而组合成一个新的字符串，代码如下：

```
<?php
$str="PHP编程宝典*NET编程宝典*ASP编程宝典*JSP编程宝典";   //定义字符串常量
$str_arr=explode("*",$str);                              //应用标识*分割字符串
$array=implode("*",$str_arr);                            //将数组合成字符串
echo $array;                                             //输出字符串
?>
```

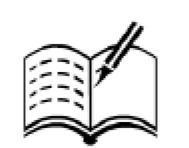
运行结果为：PHP 编程宝典*NET 编程宝典*ASP 编程宝典*JSP 编程宝典

上机演练

上机演练 1　查看最便宜的图书

在图书管理系统中，将图书的名称和价格存放在数组中，并按价格由高到低的顺序排列。当需要查找最便宜的图书时，单击“查看最便宜的图书”按钮，应用 array_pop()函数弹出数组中末尾的单元。运行结果如图 5.14 所示。







上机演练 2　随机抽取图书

应用 array_rand()函数从图书馆的技术类图书中随机抽取 3 本书作为推荐图书，运行结果如图 5.15 所示。



图 5.14　检索最便宜的图书

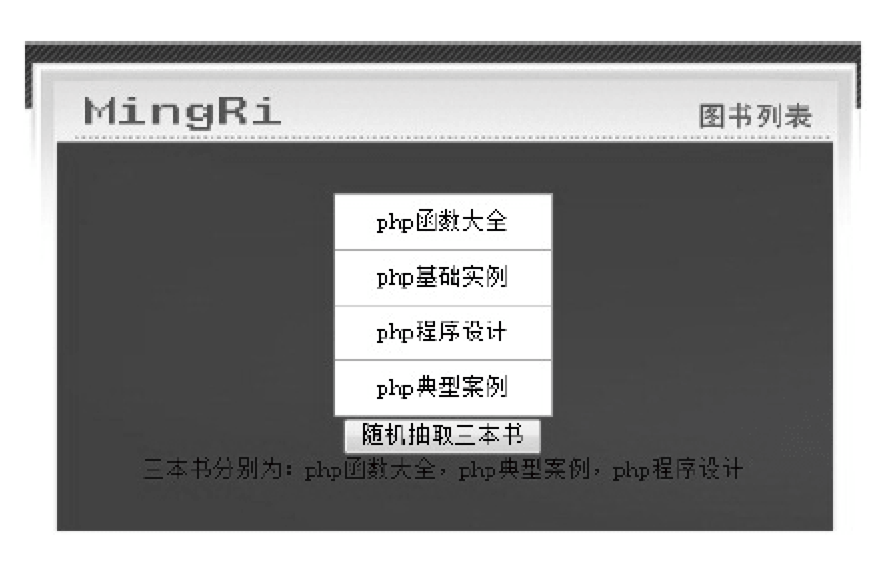


图 5.15　随机抽取图书

上机演练 3　车牌摇号

我们可以把随机抽取数组中的元素看成车牌摇号的大摇箱，在原理上两者是没什么区别的。通过随机函数 rand()实现随机抽取车牌号码，运行结果如图 5.16 所示。

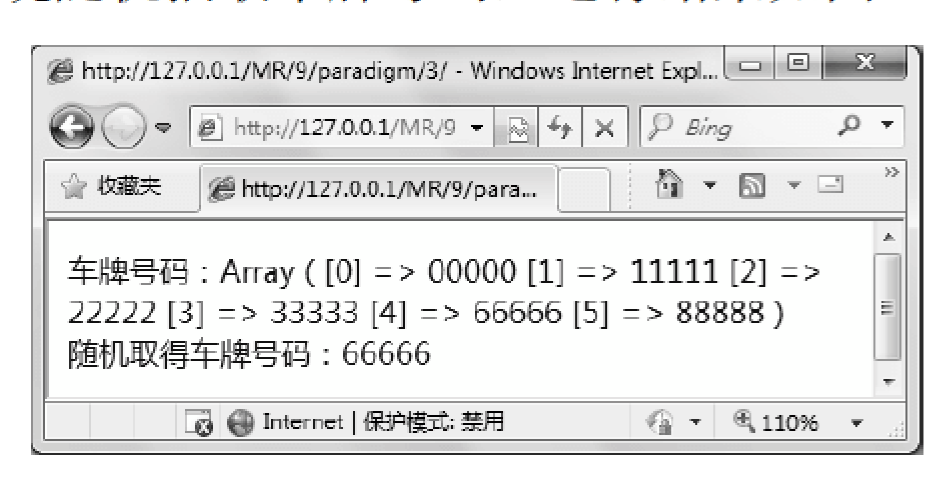


图 5.16　随机抽取车牌号码

本 章 摘 要

1．数组的类型、声明和遍历数组的方法。

2．PHP 常用数组函数的应用，包括统计数组元素个数，向数组中添加元素，获取数组中最后一个元素，以及删除数组中的重复元素等。

习　　题

1．统计数组元素个数的函数是（　　）。

A．array　　B．count　　C．foreach　　D．list

2．创建数组的函数是（　　）。

A．array　　B．count　　C．foreach　　D．list

3．在输出数组时，echo、print 和 print_r 的区别是（　　）。

A．echo 和 print_r 只可以输出数组中的某个元素值。

B．print_r 可以输出数组，而 echo 和 print 只可以输出数组中的某个元素值

C．echo 只可以输出数组中的某个元素值，print 和 print_r 可以输出数组

D．没什么区别



4．PHP 中常用遍历数组方法有几种，分别是什么（　　）。

A．1　　foreach 循环语句

B．3　　foreach 循环语句、for 循环语句、list()函数

C．4　　foreach 循环语句、for 循环语句、list()函数和 each()函数

D．5　　foreach 循环语句、for 循环语句、list()函数和 each()函数、while 循环语句

5．删除数组中的某个元素，应使用（　　）函数。

A．count()　　B．array()　　C．list()　　D．unset()

6．下面是获取已上传文件的大小，请将下面的代码补充完整。

```
<?php
echo $_FILES["userfile"]["（　　）"];
?>
```

7．下面向数组$array 中添加名元素，请将下面的代码补充完整。

```
<?php
$array=array("长春","吉林","黑龙江");
（　　）($array,'上海');
?>
```

8．删除数组中的元素，使用（　　）函数。

9．在 PHP 中将以数字作为键名的数组称为（　　）。

10．根据数组维数的不同，可以把数组分为（　　）、（　　）和（　　）。

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 8 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　图书信息逆向输出

在图书列表中，将 php 书籍以列表的形式随机存储到数组中，如图 5.17 所示。单击“逆序排列”按钮，应用 arsort()函数按图书名称进行逆向排序（由高到低），获取的结果列表如图 5.18 所示。

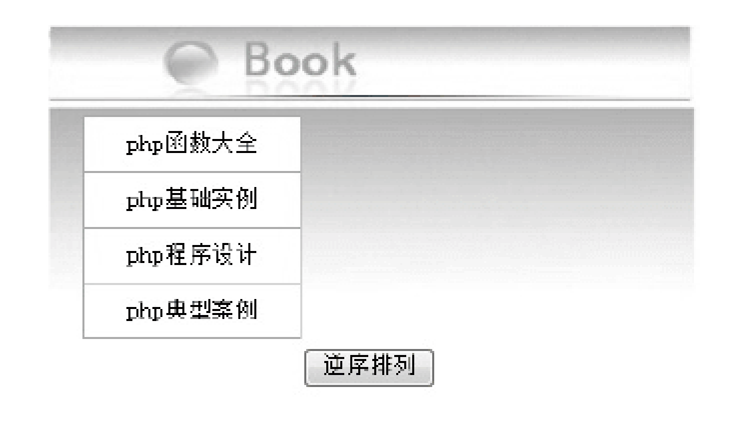


图 5.17　php 图书列表

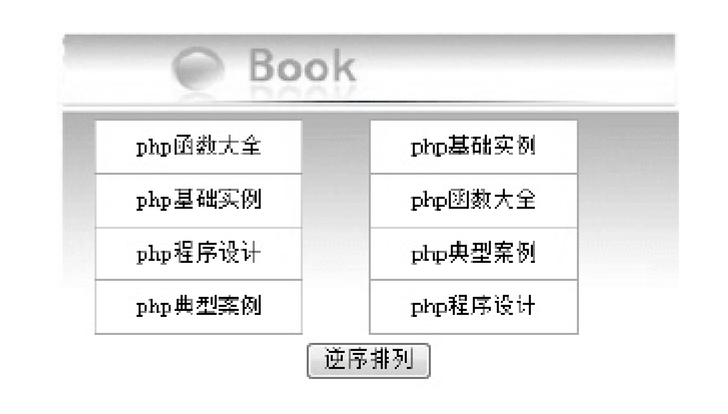


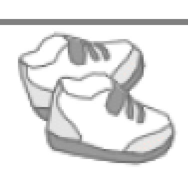
图 5.18　逆向排序后的图书列表

实战模拟 2　统计图书数量

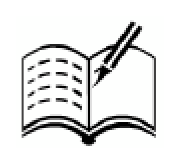
在图书统计列表页面中，将图书的数据存放在数组中，并应用 count()函数对数组中的图书数量进行统计，单击“统计图书”按钮，即可输出图书的总数量。运行结果如图 5.19 所示。

实战模拟 3　获取图书管中最受欢迎的 3 本图书

将图书馆内 php 书籍的列表及借阅人数随机存放到数组中，应用 krsort()函数对图书进行逆



向排序，然后返回排序好的数组列表，再应用 array_values()函数取出数组的值并重设数字索引，因为 list()函数仅用于数字索引并从 0 开始，最后应用 list()函数获取最受读者欢迎的 3 本 php 图书，如图 5.20 所示。





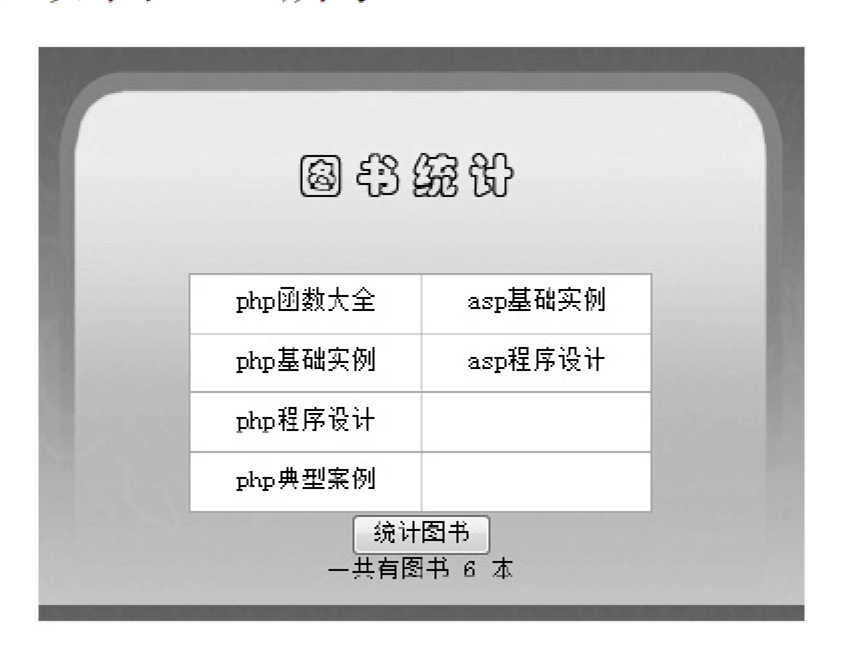


图 5.19　图书统计列表

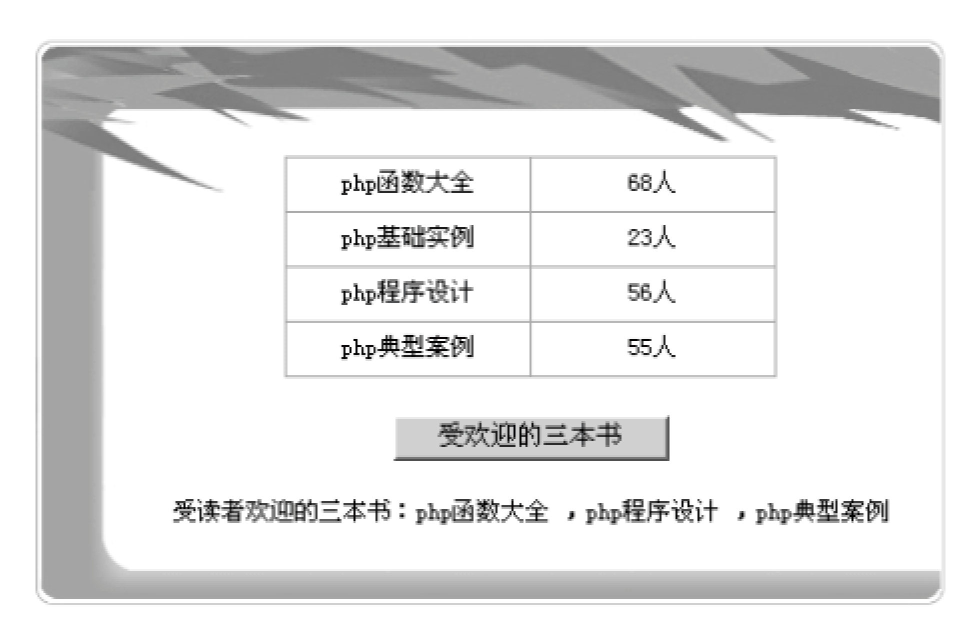


图 5.20　列举出最受读者欢迎的 3 本书

实战模拟 4　生成在线考试题

模拟在线考试的后台管理系统，完成在线考试试题的添加和输出。其运行结果如图 5.21 所示。



图 5.21　考试题输出



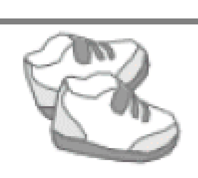
Web 技术

（自学视频、源程序：配套资源\mr\6\）

PHP 被设计为一种 Web 脚本语言，尽管它也可以用于编写命令行和图形界面程序，但在绝大多数情况下还是用于 Web 开发。动态网站可能会有表单（form）、会话（session）和重定向（redirection）等概念，本章将介绍如何在 PHP 中实现它们。

学习摘要：

- HTTP 基础
- 创建表单、表单元素设置、表单处理方法
- 获取表单参数
- 文件上传
- 表单验证
- HTML 响应头信息应用，重定向、设置过期时间和文件下载
- COOKIE 技术
- session 会话





Note

6.1 HTTP 基础

Web 的运行是基于 HTTP 协议（HyperText Transfer Protocol，超文本传输协议）的。该协议规定了浏览器如何向 Web 服务器请求文件以及服务器如何根据请求返回文件。

当一个 Web 浏览器请求一个 Web 页面时，它会发送一个 HTTP 请求消息给 Web 服务器。这个请求消息总是包含某些头部信息作为响应。回应消息也包含头部信息和消息主体。HTTP 请求的第一行通常是这样的：

```
CET /index.html HTTP/1.1
```

这一行指定一个称为方法（method）的 HTTP 命令，其后指明文档的地址和正使用的 HTTP 协议版本，意思是：通过 GET 方法进行请求，并采用 HTTP 1.1 协议来请求名称为 index.html 的服务器端文档。

在第一行之后，请求还可能包含一些可选的头部信息和服务器附加的数据。例如：

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.1.3) Gecko/20090824
Firefox/3.5.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: GB2312,utf-8;q=0.7,*;q=0.7
```

User-Agent 头提供 Web 浏览器相关的信息，而 Accept 头指定了浏览器接受的 MIME 类型。在所有头部信息之后，请求会包含一个空白行，说明头部信息已经结束；如果采用了相对应的方法（如 POST 方法），请求也可以包含附加的数据；如果不包含任何数据，它就会以一行空白行结束。

Web 服务器接收请求后，处理并返回一个响应。HTTP 响应的第一行看起来是这样的：

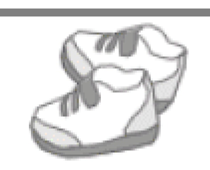
```
HTTP/1.1 200 OK
```

这一行指定协议的版本、状态码和状态码的描述。本例中状态码为“200”，说明请求成功（因此状态码的描述是“OK”）。在状态行之后，响应消息包含了一些头部信息，用于向客户端浏览器提供附加信息。例如：

```
Date: Fri, 20 May 2011 03:10:50 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_autoindex_color
PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1
Content-Length: 5197
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html;charset=UTF-8
```

Server 一行提供了 Web 服务器软件的相应信息；Content-Type 指定响应中数据的 MIME 类型。在这些信息之后是一个空白行，如果请求成功，空行之后就是所请求的数据。

最常用的两种 HTTP 方法——GET 和 POST。GET 方法用于从服务器中获得文档、图像或数据库检索结构的信息；POST 则用于向服务器发送信息，例如：信用卡的号码或其他要提交到服务器并存储到服务器上的数据库中的信息。当用户在浏览器的地址栏中输入一个 URL 并访问或者单击网页上的一个链接时，浏览器都使用 GET 方法。而用户提交一个表单时，既可以使用 POST 方法，也可以使用 GET 方法，具体使用哪种方法由 form 标签的 method 属性确



定，关于 GET 和 POST 方法将在 6.4 节中进行详细介绍。



☎小测试

1. HTTP 协议请求和响应过程

HTTP 协议采用请求/响应模型。客户端向服务器发送一个请求，请求头包含请求的方法、URI、协议版本，以及包含请求修饰符、客户信息和内容类似于 MIME 的消息结构。服务器以一个状态行作为响应，相应的内容包括消息协议的版本，成功或者错误编码加上包含服务器信息、实体元信息以及可能的实体内容。HTTP 协议请求和响应示例如下：

```
http://192.168.1.59/TM/

GET /TM/ HTTP/1.1
Host: 192.168.1.59
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.1.3) Gecko/20090824
Firefox/3.5.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: GB2312,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

HTTP/1.1 200 OK
Date: Fri, 20 May 2011 03:10:50 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_autoindex_color
PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1
Content-Length: 5197
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html;charset=UTF-8
```

2. 服务器的响应

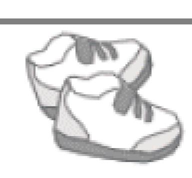
一个响应由四个部分组成：状态行、响应头标、空行、响应数据。

（1）状态行：状态行由以下三个标记组成。

- ☑ HTTP 版本：向客户端指明其可理解的最高版本。例如：HTTP/1.1 200 OK。
- ☑ 响应代码：3 位的数字代码，指出请求的成功或失败，如果失败则指出原因。
  - ➢ 1xx：信息，请求收到，继续处理。
  - ➢ 2xx：成功，行为被成功地接受、理解和采纳。
  - ➢ 3xx：重定向，为了完成请求，必须进一步执行的动作。
  - ➢ 4xx：客户端错误。
- ☑ 响应描述：为响应代码的可读性解释。

（2）响应头标：像请求头标一样，它们指出服务器的功能，标识出响应数据的细节。

（3）空行：最后一个响应头标之后是一个空行，发送回车符和退行，表明服务器以下不



再有头标。

（4）响应数据：HTML 文档和图像等，也就是 HTML 本身。

3. HTTP 规范的请求方法



Note

HTTP 规范定义了 8 种可能的请求方法：

- ☑ GET：检索 URI 中标识资源的一个简单请求。
- ☑ HEAD：与 GET 方法相同，服务器只返回状态行和头标，并不返回请求文档。
- ☑ POST：服务器接受被写入客户端输出流中的数据的请求。
- ☑ PUT：服务器保存请求数据作为指定 URI 新内容的请求。
- ☑ DELETE：服务器删除 URI 中命名的资源的请求。
- ☑ OPTIONS：关于服务器支持的请求方法信息的请求。
- ☑ TRACE：Web 服务器反馈 HTTP 请求和其头标的请求。
- ☑ CONNECT：已文档化但当前未实现的一个方法，预留做隧道处理。

6.2 变　　量

视频讲解：配套资源\mr\6\video\变量.exe

这里讲解的不是 PHP 脚本中的普通变量，而是 PHP 脚本中获取服务器环境信息、请求信息（包括表单参数和 Cookie）的方法。通常把这些信息统称为 EGPCS（environment、GET、POST、Cookies、server）。

如果 php.ini 文件中的 register_globals 选项被启用，PHP 就会为第一个表单参数、请求信息服务器环境创建一个独立的全局变量。该功能非常方便，它可以让浏览器为程序初始化任何变量，但其也非常危险，这一点将在后面的章节中介绍。

如果忽略 register_globals 的设置，PHP 将创建 6 个包含 EGPCS 信息的全局数组，通过它们获取 EGPCS 传递的数据。

- ☑ S_COOKIE：获取 COOKIE 中传递的所有 Cookie 值，数组的键名是 Cookie 名称。
- ☑ S_GET：获取 GET 请求传递的参数值，数组的键名是表单参数的名称。
- ☑ S_POST：获取 POST 请求传递的参数值，数组的键名是表单参数的名称。
- ☑ S_FILES：获取上传文件的所有信息。
- ☑ S_SERVER：获取服务器的相关信息。
- ☑ S_ENV：获取环境变量的值，键名是环境变量的名称。

这些变量不但是全局的，而且在函数的定义中也是可见的。S_REQUEST 数组也由 PHP 自动生成，包含了 S_GET、S_POST、S_COOKIE 3 个数组的所有元素。

另外，PHP 还会创建一个 S_SERVER['PHP_SELF']的变量，用于存放当前脚本的路径和名称（相对于文档根目录，例如/06/stat.php）。

☎小测试

1．创建一个表单，设置 method 属性值为 GET，在表单中添加一个文本框，设置 name 属性为 user，如何获取表单元素的值（　　）。



A．$_GET["user"]　　B．$_POST["user"]
C．$_REQUEST ["user"]　　D．$_SERVER["user"]

2．创建一个表单，设置 method 属性值为 POST，在表单中添加一个文本框，设置 name 属性为 user，如何获取表单元素的值（　　）。

A．$_GET["user"]　　B．$_POST["user"]
C．$_REQUEST ["user"]　　D．$_SERVER["user"]



3．创建一个超链接，并且为其设置一个参数 subject_id，参数值为 8。创建超链接代码如下：帖子内容。那么该如何获取超链接传递的参数值（　　）。

A．$_GET["subject_id "]　　B．$_POST["subject_id "]
C．$_REQUEST ["subject_id "]　　D．$_SERVER["subject_id "]

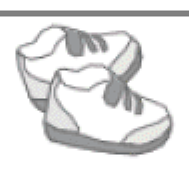
6.3　服务器信息

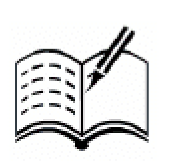
$_SERVER 数组包含很多服务器相关的有用信息，其中大部分信息来自于 CGI 规范（http://hoohoo.ncsa.uiuc.edu/cgi/env.html）中所要求的环境变量（environment variable）。

下面对$_SERVER[ ]全局数组进行介绍，如表 6.1 所示。

表 6.1　$_SERVER[ ]全局数组

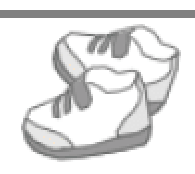
参　　数	说　　明
$_SERVER['DOCUMENT_ROOT']	当前运行脚本所在的文档根目录。在服务器配置文件中定义
$_SERVER['HTTP_HOST']	当前请求的 Host: 头信息的内容
$_SERVER['PHP_SELF']	当前正在执行脚本的文件名，与 document root 相关。例如：URL 地址为 http://example.com/test.php/foo.bar 的脚本中使用$_SERVER['PHP_SELF']将会得到/test.php/foo.bar 这个结果
$_SERVER['REMOTE_ADDR']	请求本页的机器 IP 地址，如“192.168.1.59”
$_SERVER['REQUEST_URI']	访问此页面所需的 URI，如“/index.html”
$_SERVER['SERVER_NAME']	主机名、DNS 别名或 IP 地址，如“www.example.com”
$_SERVER['SERVER_SIGNATURE']	包含服务器版本和虚拟主机名的字符串
$_SERVER ['argv']	传递给该脚本的参数。当脚本运行在命令行方式时，argv 变量传递给程序 C 语言样式的命令行参数。当调用 GET 方法时，该变量包含请求的数据
$_SERVER[' argc ']	包含传递给程序的命令行参数的个数（如果运行在命令行模式）
$_SERVER['GATEWAY_INTERFACE']	所遵循的 CGI 标准的版本号，如“CGI/1.1”
$_SERVER ['SERVER_SOFTWARE']	一个用于标识服务器的字符串，如“Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1”

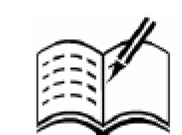




续表

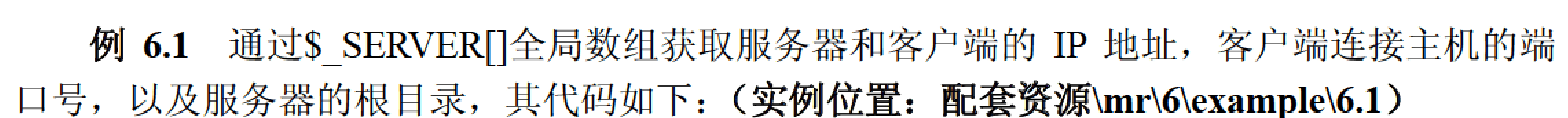
参　　数	说　　明
$_SERVER['SERVER_PROTOCOL ']	请求页面时通信协议的名称和版本，如“HTTP/1.0”
$_SERVER['REQUEST_METHOD']	客户端获取文档的方法，如“GET”
$_SERVER['REQUEST_TIME']	请求开始时的时间戳。从 PHP 5.1.0 起有效
$_SERVER['QUERY_STRING']	检索字符串，URL 中问号？后面的部分，例如 URL 为“index.php?subject_id=35”，则$_SERVER['QUERY_STRING']为“subject_id=35”
$_SERVER['HTTP_ACCEPT']	当前请求的 Accept: 头信息的内容
$_SERVER['HTTP_ACCEPT_CHARSET']	当前请求的 Accept-Charset: 头信息的内容，如“iso-8859-1,*,utf-8”
$_SERVER['HTTP_ACCEPT_ENCODING']	当前请求的 Accept-Encoding: 头信息的内容，如“gzip”
$_SERVER['HTTP_ACCEPT_LANGUAGE']	当前请求的 Accept-Language: 头信息的内容，如“en”
$_SERVER['HTTP_CONNECTION']	当前请求的 Connection: 头信息的内容，如“Keep-Alive”
$_SERVER['HTTP_REFERER']	浏览器来到当前页面的上一个页面，如“http://www.example.com/last_page.html”
$_SERVER['HTTP_USER_AGENT']	标识浏览器的字符器，如“Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1)”，表示访问当前页面的用户使用的操作系统、浏览器类型和版本等
$_SERVER['HTTPS']	如果脚本是通过 HTTPS 协议被访问，则被设为一个非空的值
$_SERVER['REMOTE_HOST']	请求本页的机器主机名，如 dialup-192-168-0-1.example.com。如果机器没有 DNS 记录，则这个变量为空，只给出 REMOTE_ADDR 值
$_SERVER['REMOTE_PORT']	用户连接到服务器时所使用的端口
$_SERVER['SCRIPT_FILENAME']	当前执行脚本的绝对路径名 注：如果脚本在 CLI 中被执行，作为相对路径，例如 file.php 或 ../file.php，$_SERVER['SCRIPT_FILENAME'] 将包含用户指定的相对路径
$_SERVER['SERVER_ADMIN']	该值指明了 Apache 服务器配置文件中的 SERVER_ADMIN 参数。如果脚本运行在一个虚拟主机上，则该值是那个虚拟主机的值
$_SERVER['SERVER_PORT']	请求发送到的服务器端口号，如“80”
$_SERVER['PATH_TRANSLATED']	PATH_INFO 的值，由服务器转换成文件名，如“/home/httpd/htdocs/list/users”
$_SERVER['SCRIPT_NAME']	当前页面的 URL 路径，用于自引用脚本，如“/~me/menu.php”
$_SERVER['PHP_AUTH_DIGEST']	当作为 Apache 模块运行时，进行 HTTP Digest 认证的过程中，此变量被设置成客户端发送的“Authorization” HTTP 头内容（以便进行进一步的认证操作）
$_SERVER['PHP_AUTH_USER']	当 PHP 运行在 Apache 或 IIS（PHP 5 是 ISAPI）模块方式下，并且正在使用 HTTP 认证功能，这个变量便是用户输入的用户名





续表

参　　数	说　　明
$_SERVER['PHP_AUTH_PW']	当 PHP 运行在 Apache 或 IIS（PHP 5 是 ISAPI）模块方式下，并且正在使用 HTTP 认证功能，这个变量便是用户输入的密码
$_SERVER['AUTH_TYPE']	如果本页面受到密码保护，则此变量指明了保护本页面的验证方法，如“basic”



例 6.1　通过$_SERVER[]全局数组获取服务器和客户端的 IP 地址，客户端连接主机的端口号，以及服务器的根目录，其代码如下：（**实例位置：配套资源\mr\6\example\6.1**）

```
<?php
    echo "当前服务器IP地址是：<b>".$_SERVER['SERVER_ADDR']."</b><br>";
    echo "当前服务器的主机名称是：<b>".$_SERVER['SERVER_NAME']."</b><br>";
    echo "客户端IP地址是：<b>".$_SERVER['REMOTE_ADDR']."</b><br>";
    echo "客户端连接到主机所使用的端口：<b>".$_SERVER['REMOTE_PORT']."</b><br>";
    echo "当前运行的脚本所在文档的根目录：<b>".$_SERVER['DOCUMENT_ROOT'].
"</b><br>";
?>
```

运行结果如图 6.1 所示。

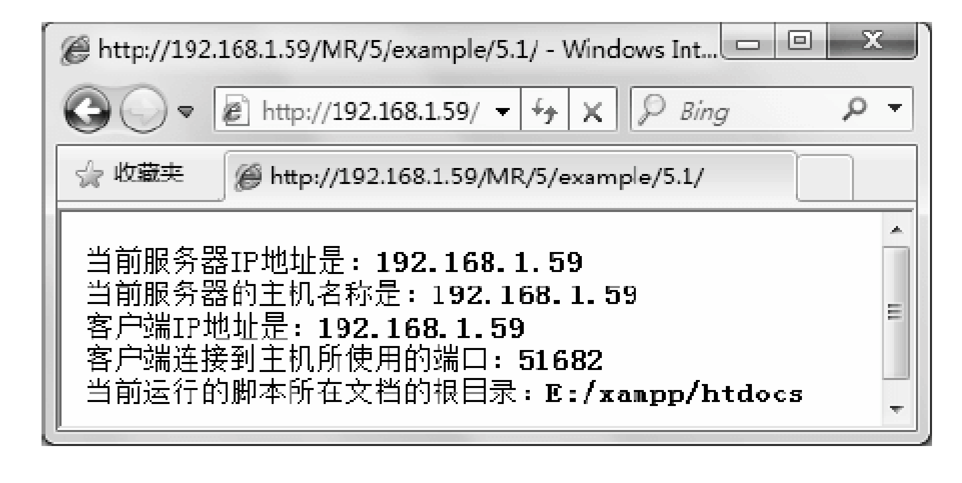


图 6.1　获取服务器和客户端的 IP 地址

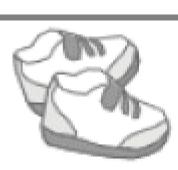
上机演练

上机演练 1　通过客户端 IP 限制投票次数

下面制作一个简单的投票系统，通过获取客户端 IP 地址来限制用户的投票次数，每个 IP 只可以投票一次，如果重复投票将给出提示。运行结果如图 6.2 所示。



图 6.2　在线投票



获取客户端 IP 地址需要使用$_SERVER[]全局数组中的$_SERVER['REMOTE_ADDR']参数。本例将客户端 IP 地址存放到数据库中，当用户提交投票时，使用 if...else 语句判断该 IP 在数据库中是否存在，从而实现通过客户端 IP 地址限制投票次数的目的。



Note

6.4 表单处理

视频讲解：配套资源\mr\6\video\表单处理.exe

因为表单变量可以通过$_GET 和$_POST 数组得到，所以用 PHP 处理表单十分容易。尽管如此，表单处理还是有很多技巧的，本节将进行详细介绍。

6.4.1 创建表单

表单是使用<form></form>标签来创建并定义开始和结束位置，中间包含多个元素，其结构如下：

```
<form name="form_name"  method="method" action="url"  enctype="value"  target="target_win" id="id">
……                                    //省略插入的表单元素
</form >
```

<form>标记的属性如表 6.2 所示。

表 6.2　表单的常用属性

属　　性	描　　述
name	表单名称
id	表单的 ID 号
method	该属性用于定义表单中数据的提交方式，可取值为 GET 和 POST 中的一个。GET 方法将表单内容附加在 URL 地址后面进行提交，所以对提交信息的长度进行了限制，不可以超过 8192 个字符，同时 GET 方法不具有保密性，不适合处理如信用卡卡号等要求保密的内容，而且不能传送非 ASCII 的字符；POST 方法将用户在表单中填写的数据包含在表单的主体中，一起传送到服务器，不会在浏览器的地址栏中显示，这种方式传送的数据没有大小限制。默认为 GET 方法
action	该属性定义将表单中的数据提交到哪个文件中进行处理，这个地址可以是绝对 URL，也可以是相对 URL。如果该属性是空值则提交到当前文件
enctype	设置表单资料的编码格式
target	该属性和链接中的同名属性类似，用于指定目标窗口或目标帧

6.4.2 添加表单元素

表单（form）由表单元素组成。常用的表单元素有以下几种标记：输入域标记<input>、选择域标记<select>和<option>、文本域标记<textarea>等。下面分别进行介绍。

1. 输入域标记<input>

输入域标记<input>是表单中最常用的标记之一。常用的文本域、按钮、单选复选框等构成了一个完整的表单。语法如下：



```
<form>
<input name="filed_name"   type="type_name">
</form>
```

参数 name 是指输入域的名称；参数 type 是指输入域的类型。在<input type="">标记中一共提供了 10 种类型的输入区域，用户所选择使用的类型由 type 属性决定。type 属性取值及举例如表 6.3 所示。

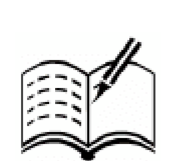


表 6.3 type 属性取值及举例

值	举 例	说 明	运 行 结 果
text	<input name="user" **type="text"** value=" 纯净水" size="12" maxlength="1000">	name 为文本框的名称，value 是文本框的默认值，size 指文本框的宽度（以字符为单位），maxlength 指文本框的最大输入字符数	添加一个文本框： 纯净水
hidden	<input **type="hidden"** name="ddh">	隐藏域，用于在表单中以隐含方式提交变量值。隐藏域在页面中对于用户而言是不可见的，添加隐藏域的目的在于通过隐藏的方式收集或发送信息。当浏览者单击发送按钮发送表单时，隐藏域的信息也被一起发送到 action 指定的处理页	添加一个隐藏域：
password	<input name="pwd" **type= "password"** value="666666" size= "12" maxlength="20">	密码域，用户在该文本框输入字符时将被替换显示为 * 号，起到保密作用	添加一个密码域： ******
file	<input name="file" **type="file" enctype="multipart/form-data"** size="16" maxlength="200">	文件域，当文件上传时，可用来打开一个模式窗口以选择文件。然后将文件通过表单上传到服务器，如上传 Word 文件等各种类型的文件。但是必须注意的是，在上传文件时需要指明表单的属性 enctype="multipart/form-data"才可以实现上传功能	添加一个文件域： 浏览...
image	<input name="imageField" **type= "image"** src="images/banner.gif" width="120" height="24" border= "0">	图像域是指可以用在提交按钮位置上的图片，这幅图片具有按钮的功能	添加一个图像域：
radio	<input name="sex" **type="radio"** value="1" **checked**>男 <input name="sex" **type="radio"** value="0">女	单选按钮，用于设置一组选择项，用户只能选择一项。checked 属性用来设置单选按钮默认值	添加一组单选按钮（如您的性别为：） ◉ 男 ○ 女
checkbox	<input name="checkbox" **type= "checkbox"** value="1" **checked**> 封面 <input name="checkbox" **type= "checkbox"** value="1" **checked**> 正文内容 <input name="checkbox" **type= "checkbox"** value="0">价 格	复选框，允许用户同时选择多个选择项。checked 属性用来设置该复选框默认值。例如，在收集个人信息时，要求在个人爱好的选项中进行多项选择等	添加一组复选框（如影响您购买本书的因素：） ☑ 封面 ☑ 正文内容 ☐ 价 格





续表

值	举　　例	说　　明	运 行 结 果
submit	<input **type="submit"** name="Submit" value="提交">	将表单的内容提交到服务器端	添加一个提交按钮： 提交
reset	<input **type="reset"** name="Submit" value="重置">	清除与重置表单内容，用于清除表单中所有文本框的内容，而且使选择菜单项恢复到初始值	添加一个重置按钮： 重置
button	<input **type="button"** name="Submit" value="按钮">	按钮可以激发提交表单的动作，可以在用户需要修改表单时，将表单恢复到初始的状态，同时还可以依照程序的需要，发挥其他作用。普通按钮一般是配合 JavaScript 脚本来进行表单的处理	添加一个普通按钮： 按钮

2. 选择域标记<select>和<option>

通过选择域标记<SELECT>和<OPTION>可以建立一个列表或菜单。菜单节省空间，正常状态下只能看到一个选项，单击按钮打开菜单后才能看到全部选项。列表可以显示一定数量的选项，如果超出了这个数量，就会自动出现滚动条，浏览者可以通过拖动滚动条来查看各选项。语法如下：

```
<select name="name" size="value" multiple>
<option value="value" selected>选项1</option>
<option value="value">选项2</option>
<option value="value">选项3</option>
…
</select>
```

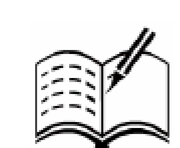
参数 name 表示选择域的名称；参数 size 表示列表的行数；参数 value 表示菜单选项值；参数 multiple 表示以菜单方式显示数据，省略则以列表方式显示数据。

选择域标记<select>和<option>的显示方式及举例如表 6.4 所示。

表 6.4　选择域标记<select>和<option>的显示方式及举例

显 示 方 式	举　　例	说　　明	运 行 结 果
列表方式	<**select** name="spec" id="spec"> <**option** value="0" **selected**>网络编程</**option**> <**option** value="1">办公自动化</**option**> <**option** value="2">网页设计</**option**> <**option** value="3">网页美工</**option**> </**select**>	下拉列表框，通过选择域标记<select>和<option>建立一个列表，列表可以显示一定数量的选项，如果超出了这个数量，会自动出现滚动条，浏览者可以通过拖动滚动条来查看各选项。selected 属性用来设置该菜单项时默认被选中	请选择所学专业： 网络编程 网络编程 办公自动化 网页设计 网页美工





续表

显示方式	举例	说明	运行结果
菜单方式	<**select** name="spec" id="spec" **multiple**> <**option** value="0" **selected**>网络编程</**option**> <**option** value="1">办公自动化</**option**> <**option** value="2">网页设计</**option**> <**option** value="3">网页美工</**option**> </**select**>	multiple 属性用于下拉列表<select>标记中，指定该选项用户可以使用 Ctrl 和 Shift 键进行多选	请选择所学专业： 网络编程 办公自动化 网页设计 网页美工



说明：

表 6.4 中给出了静态菜单项的添加方法，而在 Web 程序开发过程中，也可以通过循环语句动态添加菜单项。

3. 文本域标记<TEXTAREA>

文本域标记<TEXTAREA>用来制作多行的文本域，可以在其中输入更多的文本。语法如下：

```
<textarea name="name" rows=value cols=value value="value" warp="value">
    …文本内容
</textarea>
```

参数 name 表示文本域的名称；rows 表示文本域的行数；cols 表示文本域的列数（这里的 rows 和 cols 以字符为单位）；value 表示文本域的默认值。warp 用于设定显示和送出时的换行方式，值为 off 表示不自动换行；值为 hard 表示自动按回车键换行，换行标记一同被发送到服务器，输出时也会换行；值为 soft 表示自动按回车键换行，换行标记不会被发送到服务器，输出时仍然为一列。

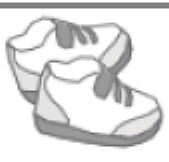
文本域标记<TEXTAREA>的值及举例如表 6.5 所示。

表 6.5 文本域标记<TEXTAREA>的值及举例

值	举例	说明	运行结果
textarea	<**textarea** name="remark" cols="20" rows="4" id="remark"> 请输入您的建议!</**textarea**>	文本域，也称多行文本框，用于多行文本的编辑默认 warp 属性时，默认为自动换行方式	请发表您的建议： 请输入您的建议!

例 6.2 创建一个表单，表单元素包含文本域、单选按钮、复选框、下拉列表和提交按钮等，具体代码如下：（**实例位置：配套资源\mr\6\example\6.2**）

```
<form id="form1" name="form1" method="post" action="">
  <table width="286" border="0" align="center">
    <tr>
      <td width="72"><span class="STYLE1">用户名：</span></td>
```




```
            <td width="204"><label>
              <input type="text" name="textfield" />
            </label></td>
          </tr>
          <tr>
            <td><span class="STYLE1">密码：</span></td>
            <td><label>
              <input type="password" name="textfield2" />
            </label></td>
          </tr>
          <tr>
            <td><span class="STYLE1">性别：</span></td>
            <td><label>
              <input name="radiobutton" type="radio" value="radiobutton" checked="checked" />
              <span class="STYLE1">男          </span>
              <input type="radio" name="radiobutton" value="radiobutton" />
            <span class="STYLE1">女</span></label></td>
          </tr>
         <!--省略部分代码-->
          <tr>
            <td colspan="2" align="center"><label>
              <input type="submit" name="Submit" value="提交" />
            </label>
              <label>
              <input type="submit" name="Submit2" value="取消" />
              </label></td>
          </tr>
        </table>
        <p>
          <label></label>
        </p>
      </form>
```

上述表单创建的界面效果如图 6.3 所示。

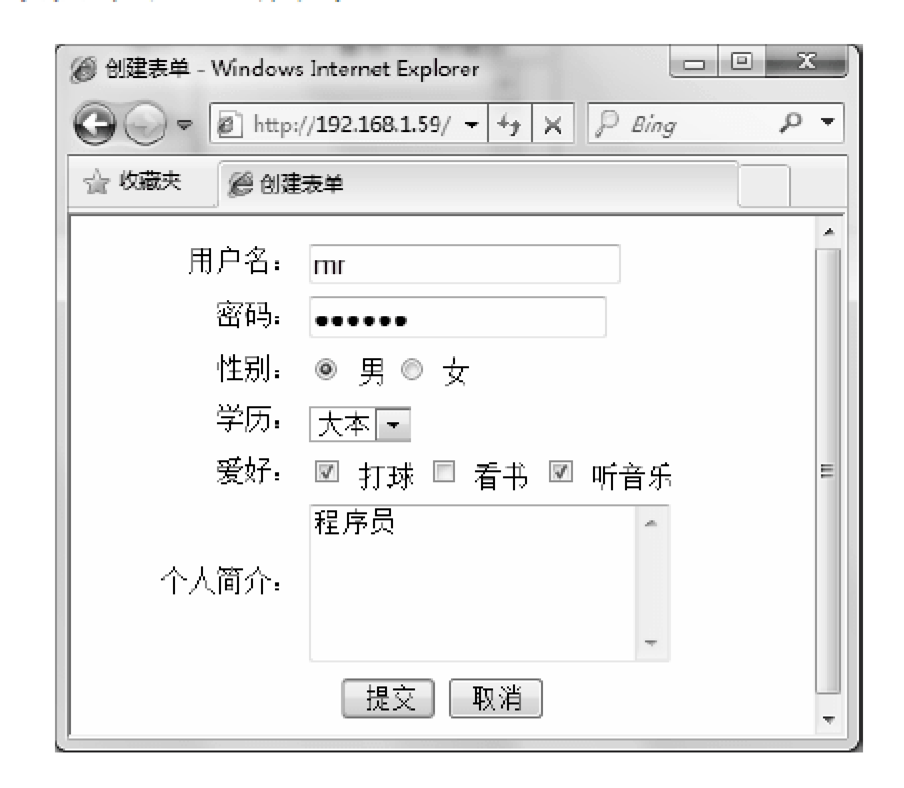
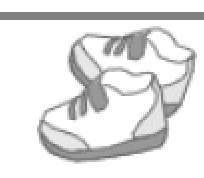
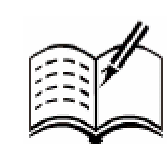


图 6.3　表单创建的界面效果





6.4.3 方法

客户端可以用两种 HTTP 方法向服务器传送表单数据：GET 和 POST。采用哪种方法是由表单标签（<form>）中的 method 属性所指定的。理论上说在 HTML 中 method 是不区分大小写的，但是实际上有些浏览器要求 method 为大写。

GET 请求把表单的参数编码成 URL 形式，称为查询字符串（query string）：

```
/path/to/index.php?subject=despicable&length=3
```

POST 提求则通过 HTTP 请求的主体来传递表单参数，不需要考虑 URL。

GET 和 POST 方法的最明显区别是 URL 行。因为 GET 请求的所有表单参数都编码在 URL 中，用户可以把一个 GET 请求加入浏览器收藏夹，而对 POST 请求却无法这样做。

GET 和 POST 请求之间的最大不同是相当微妙的。HTTP 规范指明 GET 请求是幂等的（idempotent）。也就是说，对于一个特定 URL 的 GET 请求（包含表单参数），与对应于这一特定 URL 的两个或多个 GET 请求是一样的。因此，Web 浏览器可以把 GET 请求得到的响应页面缓存起来。这是因为不管页面被请求了多少次，响应页面都是不变的。正因为幂等性，GET 请求中用于那些响应页面永不改变的性况，例如将一个单词分解成小块，或者对数字进行乘法运算等。

POST 请求不具幂等性，这意味着它们无法被缓存，在每次刷新页面时，都会重新连接服务器。当显示或刷新页面时，用户可能会看到浏览器提示“Repost form data?(重新发送表单数据)”。所以 POST 适用于响应内容可能会随时间改变的情况，例如，显示购物车的内容或在一个论坛中显示当前主题。

现实中，幂等性常常被忽略。目前浏览器的缓存功能都很差，并且“刷新”按钮很容易被用户单击到，所以程序员通常只考虑是否想将参数显示在浏览器的 URL 地址栏上，如果不想显示，就用 POST 方法。但读者要记住的是，在服务器的响应页面可能会变化的情况下（例如下订单或更新数据库），不要使用 GET 方法。

例 6.3 开发一个实例，用于获取用户的登录信息和超链接传递的参数值。具体步骤如下。**（实例位置：配套资源\mr\6\example\6.3）**

（1）创建 index.php 文件，同时定义两个 form 表单，分别使用 GET 和 POST 方法提交数据，即通过 GET 方法提交的数据传递到 get.php 文件中，而通过 POST 方法提交的数据传递到 post.php 文件中。

（2）在 index.php 文件中，创建一个超链接链接到 index.php 页，为该超链接设置一个参数 res，设置参数值为明日科技，通过 urlencode()函数对参数值进行编码。在本页中通过 isset()函数验证$_GET['res']是否存在，如果存在则将该值赋给变量$res，否则为变量$res 赋值为空。其关键代码如下：

```
<a href="index.php?res=<?php echo urlencode('明日科技');?>">$_GET[]方法获取超链接传递的参数值</a>
<p align="center" class="STYLE1">
<?php
$res=(isset($_GET['res']))?$_GET['res']:""; //检测超链接参数值是否存在
echo "获取超链接传递的参数值：".$res;
?>
```

运行结果如图 6.4 所示。



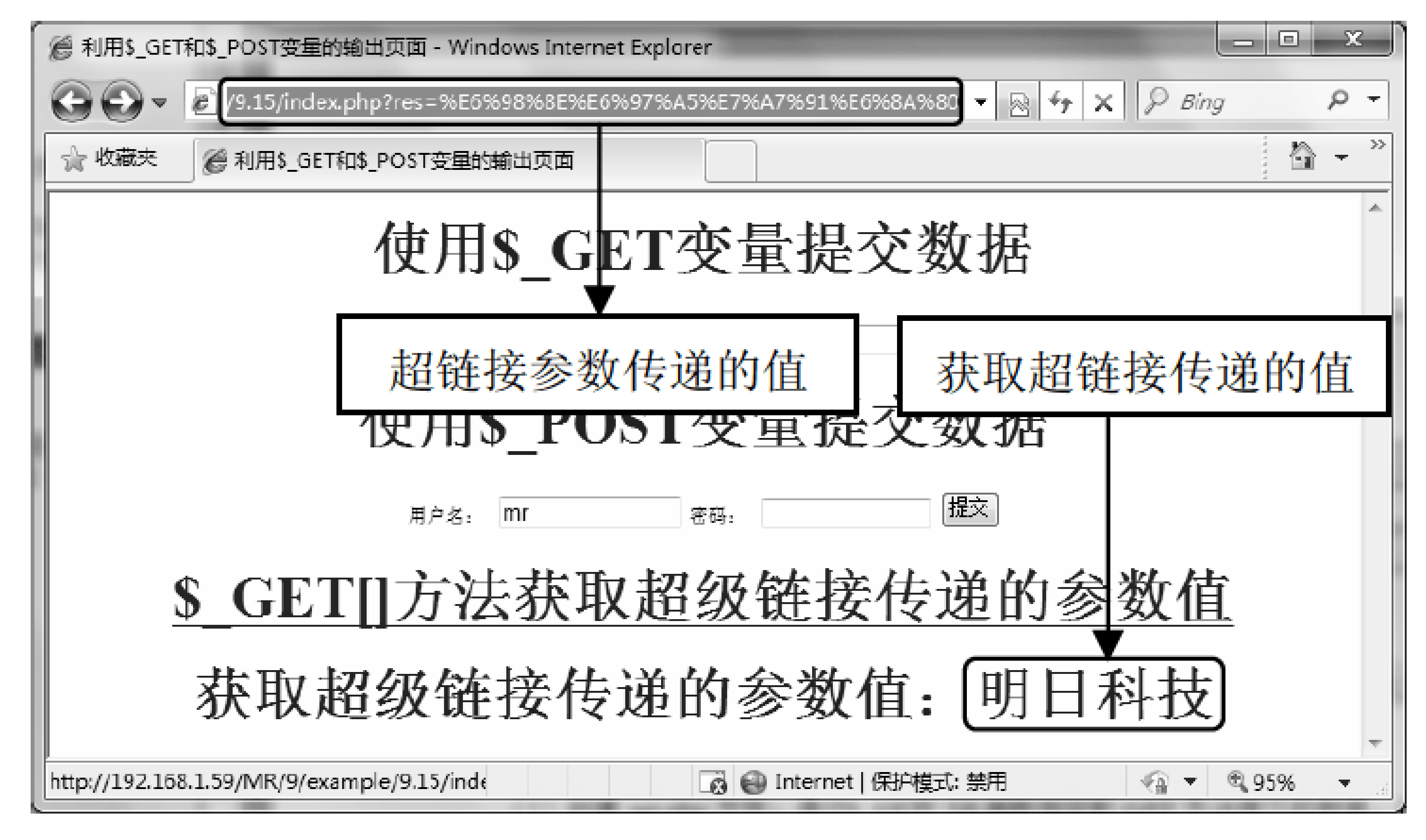



图 6.4　利用$_GET 变量的输出页面

（3）创建 get.php 文件，通过$_GET[]全局数组获取 GET 方法提交的数据。运行结果如图 6.5 所示。



图 6.5　利用$_GET 变量的输出页面

其代码如下：

```
<?php
if(isset($_GET['Submit']) and $_GET['Submit']=="提交"){
    echo "用户名为：".$_GET['user']."<br>";
    echo "密码为：".$_GET['pass'];
}
?>
```

（4）创建 post.php 文件，通过$_POST[]全局数组获取 POST 方法提交的数据，运行结果如图 6.6 所示。

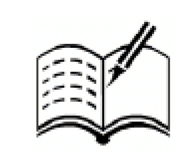


图 6.6　利用$_POST 变量的输出页面

具体代码如下：

```
<?php
if(isset($_POST['user']) and $_POST['Submit2']=="提交"){
    echo "用户名为：".$_POST['user']."<br>";
    echo "密码为：".$_POST['pass'];
}
?>
```





指点迷津：

在 PHP 程序中使用 S_POST、S_GET 和 S_FILES 数组来访问表单的参数。数组的键名是表单参数的名称，数组元素的值是表单参数的值。表单参数名称区分字母大小写。如果在编写 Web 程序时忽略字母大小写，那么在程序运行时将获取不到表单参数的值或弹出错误提示信息。

6.4.4　对参数进行自动引号处理

如果 php.ini 中的 magic_quotes_gpc 选项启用，那么 PHP 将在所有 cookie 数据以及 GET 和 POST 参数上自动调用 addslashes()函数。这使得在数据库查询中使用表单参数变得简单，但同时也对那些没有在数据库查询中使用的表单参数造成了麻烦，因为这需要在单引号、双引号、反斜杠和空字节等前面添加上反斜杠以进行转义。

例如，在文本框中输入 PHP'MRSOFT，并单击提交按钮，此时就会发现被分块的字符串其实是“PHP\'MRSOFT”。这就是 magic_quotes_gpc 的作用。

为了处理用户输入的字符串，可以禁用 php.ini 中的 magic_quotes_gpc 选项或者对$_GET、$_POST 和$_COOKIES 使用 stripslashes()函数进行转义还原。其代码如下：

```
$value=ini_get('magic_quotes_gpc')                  //获取配置选择的值
    ?stripslashes($_GET['word'])                    //原样返回转义的字符串
    :$_GET['word'];
```

如果需要处理大量字符串，还可以封装一个自定义函数，代码如下：

```
function raw_param ($name){
    return ini_get('magic_quotes_gpc')
        ?stripslashes($_GET[$name])
        :$_GET[$name];
}
```

指点迷津：

编写 PHP 程序时依赖 magic_quotes_gpc 将会导致出现很多问题，所以在新版的 PHP 所带的 php.ini 中 magic_quotes_gpc 默认设置为不启用。在 PHP6 中，本特性将被彻底移除。

6.4.5　自处理页面

所谓自处理页面，就是一个 PHP 页面能同时用来生成表单和处理表单。实现此功能有以下两种方法。

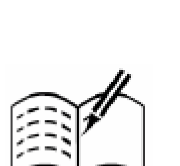
第一种方法：应用$_SERVER['REQUEST_METHOD']获取表单中 method 的值，如果值为 GET，则生成表单；如果值为 POST，则处理表单。

例 6.4　应用第一种方法完成自处理页的创建。具体步骤如下：（**实例位置：配套资源\mr\6\example\6.4**）

创建 index.php 文件，应用 if 语句判断$_SERVER['REQUEST_METHOD']获取的提交方法，如果方法为 GET，则输出创建的表单；如果方法为 POST，则获取表单提交的用户名和密码值。其关键代码如下：

```
<?php
if($_SERVER['REQUEST_METHOD']=='GET'){          //判断客户端获取文档的方法
?>
```





Note

```
<form id="form2" name="form2" method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
   <p align="center" class="STYLE1">自助处理页</p>
   <p align="center"><span class="STYLE2">用户名：</span>
        <label>
        <input name="user" type="text" size="15" id="user" />
      </label>
        <span class="STYLE2">密码：</span>
        <label>
        <input name="pass" type="password" size="15" id="pass" />
      </label>
        <label>
        <input type="submit" name="Submit" value="提交" />
        </label>
   </p>
</form>
<?php
}else if($_SERVER['REQUEST_METHOD']=='POST'){//判断客户端获取文档的方法
     echo "用户名为：".$_POST['user']."<br>";
     echo "密码为：".$_POST['pass'];
}else{
     echo "无内容";
}
?>
```

运行结果如图 6.7 所示。



图 6.7　利用$_GET 变量的输出页面

第二种方法：通过 isset()函数判断指定的参数是否被创建，如果存在则执行处理表单的操作，否则执行生成表单的操作。其关键代码如下：

```
<?php
if(!isset($_POST['user'])){          //判断参数是否被设置
?>
<form id="form2" name="form2" method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
   <p align="center" class="STYLE1">自助处理页</p>
   <p align="center"><span class="STYLE2">用户名：</span>
        <label>
        <input name="user" type="text" size="15" id="user" />
      </label>
        <span class="STYLE2">密码：</span>
```





```
            <label>
            <input name="pass" type="password" size="15" id="pass" />
        </label>
            <label>
            <input type="submit" name="Submit" value="提交" />
            </label>
    </p>
</form>
<?php
}else{
        echo "用户名为：".$_POST['user']."<br>";
        echo "密码为：".$_POST['pass'];
}
?>
```

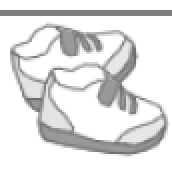
6.4.6 粘性表单

目前很多网站都使用一种称为“粘性表单”（sticky form）的技术，使用该技术可设置一个查询表单的默认值为先前查询的值。例如，如果在百度（http://www.baidu.com）上查询“明日科技”，则在结果页面的顶端的另一个查询文本框中，包含先前的查询关键字“明日科技”；如果将查询的关键字改为“明日科技 编程词典”，那么只要简单地在后面补充即可。这就是粘性表单。

例 6.5 通过粘性表单设置查询的关键字，具体步骤如下。（**实例位置：配套资源\mr\6\example\6.5**）

创建 index.php 文件，首先应用 if 语句和 isset()函数判断查询的关键字是否存在，如果存在，则将查询关键字赋给变量$key；否则为变量$key 赋空值。然后创建表单，将数据提交到本页，并将变量$key 的值作为关键字文本框的默认值，设置粘性表单。其关键代码如下：

```
<?php
if(!isset($_POST['key'])){                //判断参数是否被设置
        $key='';                          //如果为提交查询关键字，则为变量赋值为空
}else{
        $key=$_POST['key'];               //否则将提交的关键字赋给指定的变量
}
?>
<form id="form2" name="form2" method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
    <p align="center" class="STYLE1">粘性表单</p>
    <p align="center"><span class="STYLE2">关键字：</span>
            <label>
            <input name="key" type="text" size="15" id="key" value="<?php echo $key;?>" />
                </label>
            <label>
            <input type="submit" name="Submit" value="查询" />
            </label>
    </p>
</form>
 <p align="center" class="STYLE1">
```



```
<?php
    echo "查询结果:" $key."<br>";
?>
</p>
```



Note

运行结果如图 6.8 所示。



图 6.8　粘性表单应用

6.4.7　多值参数

可使用 HTML 中的 select 标签创建选择列表，允许用户进行多重选择。为了确保 PHP 识别浏览器传递来的多个值，需要在 HTML 表单的字段名后加上“[]”，例如：

```
<select name="languages[]">
    <input name="c">C</input>
    <input name="c++">C++</input>
    <input name="php">PHP</input>
    <input name="perl">Per1</input>
</select>
```

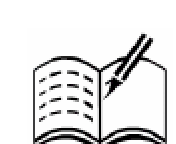
现在，当用户提交表单时，$_POST['languages']包含的是一个数组而不是一个字符串，该数组包含用户所选择的值。

例 6.6　获取表单中 select 标签提交的多值参数，其具体操作步骤如下。（**实例位置：配套资源\mr\6\example\6.6**）

创建 index.php 文件，首先应用 if 语句和 array_key_exists()函数判断提交值是否存在，如果存在，则将其赋给变量$languages；否则为变量$languages 赋空值。然后创建表单，将数据提交到本页。其关键代码如下：

```
<?php
if(!array_key_exists('Submit',$_POST)){            //判断提交的数组中是否存在该值
    $languages='';                                  //如果不存在，则为变量赋值为空
}else{
    $languages=$_POST['languages'];                 //否则将提交的数据赋给指定的变量
}
?>
<form id="form2" name="form2" method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
    <p align="center" class="STYLE1">多值参数/p>
    <p align="center"><span class="STYLE2">关键字：</span>
 <select name="languages[]" multiple>
    <option value=="c">C</input>
```






```
            <option value=="c++">C++</input>
            <option value=="php">PHP</input>
            <option value=="perl">Perl</input>
        </select>
            <label>
            <input type="submit" name="Submit" value="提交" />
            </label>
            </p>
    </form>
     <p align="center" class="STYLE1">
    <?php
        print_r($languages);
    ?> </p>
```

运行效果如图 6.9 所示。

多学两招：

上述介绍的是通过下拉列表实现多值参数传递，另外，还可以应用复选框实现多值参数传递，即将复选框（check box）的名称设置为统一的 name[]格式。

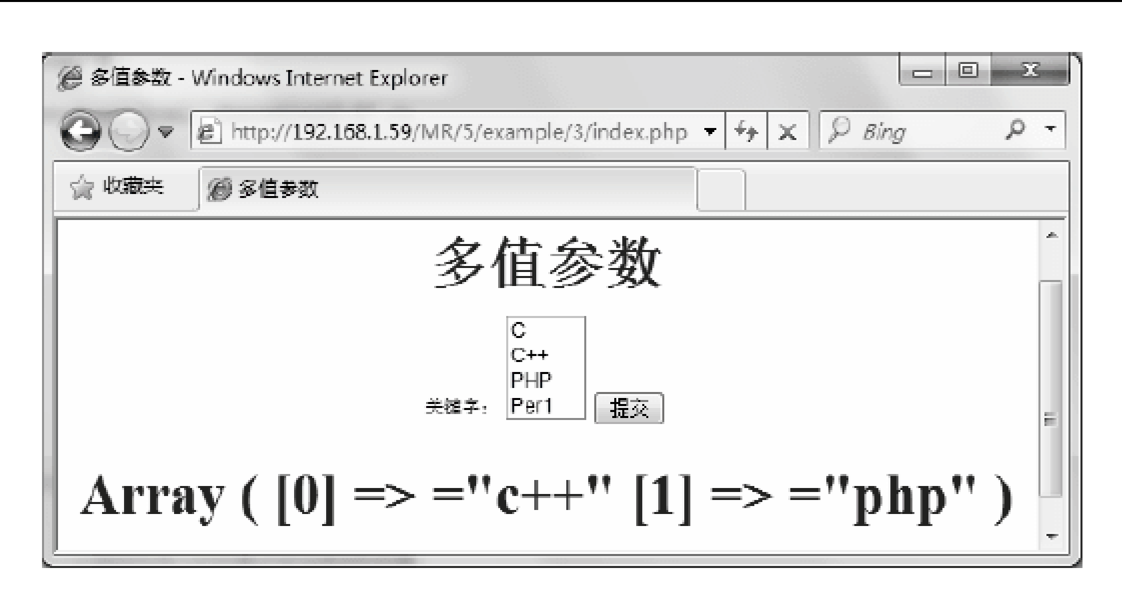


图 6.9　多值参数应用

6.4.8　粘性多值参数

前面介绍了粘性表单，那么是否可以让多值参数的表单也具有粘性呢？答案是肯定的。其方法是封装一个自定义函数，改编复选框创建的方式，以此来达到表单的粘性功能。自定义函数 make_checkbox()的语法如下：

```
function make_checkbox($name, $checked, $option){
    foreach($option as $value => $label){
        printf('%s <input type="checkbox" name="%s[]" value="%s" ', $label ,$name ,$value);
        if(in_array($value, $checked)){
            echo "checked";
        }
        echo "/>  \n";
    }
}
```

自定义函数 make_checkbox()创建具有相同名称的复选框，其中参数 name 指定复选框组的名称；参数 checked 设置复选框默认值；参数 option 定义复选框的名称和值。

例 6.7　通过自定义函数 make_checkbox()创建粘性多值表单。具体步骤如下：（**实例位置：**





Note

配套资源\mr\6\example\6.7）

创建 index.php 文件，首先判断是否执行提交操作，如果未执行，则定义空数组；否则将提交的值赋给数组。然后载入自定义函数 make_checkbox()，将复选框的名称和值定义到数组中。接着创建表单，通过自定义函数创建复选框。最后输出复选框提交的数组。其关键代码如下：

```
<?php
if(!array_key_exists('Submit',$_POST)){              //判断参数是否被设置
    $languages=array();                              //如果为空，则定义空数组
}else{
    $languages=$_POST['languages'];                  //否则将提交的值赋给指定数组
}
?>
<?php
//省略了自定义函数
//定义复选框的名称和值
$checkbox=array(
    'C' => "c",
    'C++' => "c++",
    'PHP' => "php",
    'Perl' => "perl"
);
?>
<form id="for" name="for" method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
    <p align="center" class="STYLE1">粘性多值参数</p>
    <p align="center"><span class="STYLE2">最流行的语言：</span>
<?php
make_checkbox('languages',$languages,$checkbox);
?>
     </p>
    <p align="center" class="STYLE1"><label>
        <input type="submit" name="Submit" value="提交" />
    </label></p>
</form>
<p align="center" class="STYLE1">
<?php
print_r($languages);
?>
```

运行结果如图 6.10 所示。

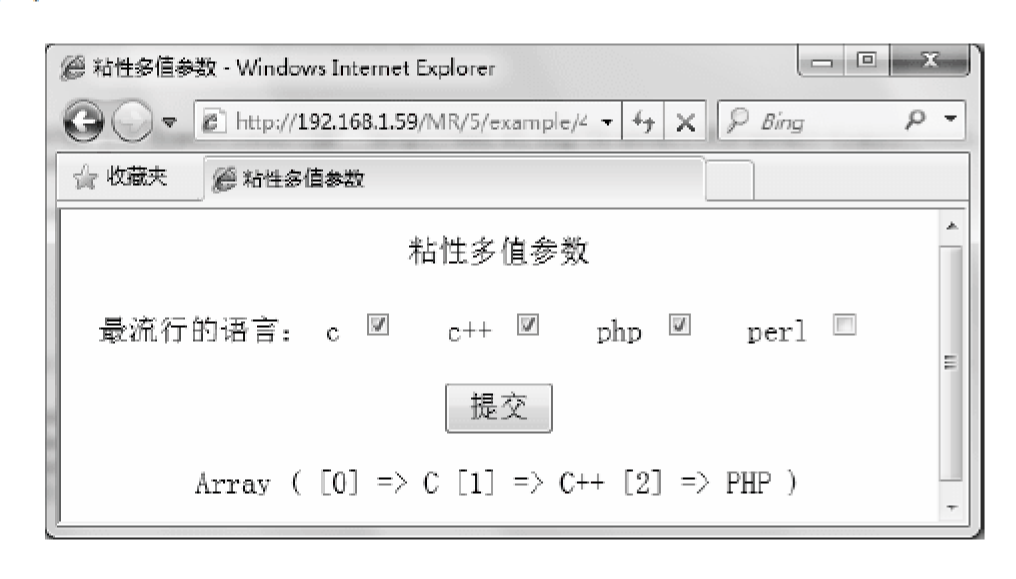
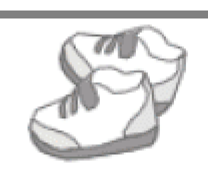


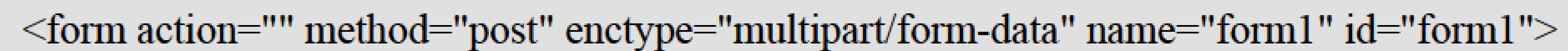
图 6.10　粘性多值参数应用



6.4.9　文件上传



文件上传，即通过表单中的文件域提交上传文件。通过$_FILES 数组处理文件，通过 is_uploaded_file()函数验证上传文件，通过 move_uploaded_file()函数完成文件上传。下面编写一个上传文件的表单，将数据提交到本页。其代码如下：



```
<form action="" method="post" enctype="multipart/form-data" name="form1" id="form1">
  <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
  <input type="file" name="fileField" id="fileField" />
  <input type="submit" name="button" id="button" value="上传" />
</form>
```

在 PHP 中上传文件的最大问题是对超大文件的处理。PHP 有两种方法避免出现这种情况：一种是硬性限制，另一种是软性限制。

（1）在 php.ini 文件中可以对上传文件进行硬性限制。包括：是否支持上传、上传文件的临时目录、上传文件的大小、指令执行的时间、指令分配的内存空间。

- ☑ file_uploads：如果值为 on，则说明服务器支持文件上传；如果值为 off，则说明不支持。一般默认为支持，这个不用修改。
- ☑ upload_tmp_dir：上传文件临时目录。在文件被成功上传之前，文件首先存放到服务器端的临时目录中。多数使用系统默认目录，但是也可以自行设置。
- ☑ upload_max_filesize：服务器允许上传文件的最大值，以 MB 为单位。系统默认为 2MB，如果需要上传超过 2MB 的数据，那么就要修改这个值。

上述是 php.ini 的 File_Uploads 项中与上传相关选项参数设置说明，除了 File_Uploads 项中的内容外，在 php.ini 中还有其他几个选项会影响到文件的上传。

- ☑ max_execution_time：PHP 中一个指令所能执行的最大时间，单位是秒。该选项在上传超大文件时必须要修改，否则即使上传文件在服务器允许的范围内，但是若超过了指令所能执行的最大时间，则仍然无法实现上传。
- ☑ memory_limit：PHP 中一个指令所分配的内存空间，单位为 MB。它的大小同样会影响到超大文件的上传。

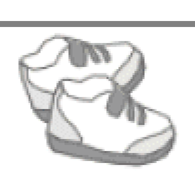
脚下留神：

（1）在 php.ini 文件配置完成后，需要重新启动 Apache 服务器，配置才能生效。（2）如果使用集成化的安装包来配置 PHP 的开发环境，那么就不必担心上述配置信息，因为默认已经配置好。（3）如果要上传超大的文件，那么就有必要对 php.ini 进行修改（即使使用集成化安装包配置的也不行）。修改选项包括：upload_max_filesize 的最大值，max_execution_ time 一个指令所能执行的最大时间和 memory_limit 一个指令所分配的内存空间。

（2）在表单中，在文件域之前添加一个名称为 MAX_FILE_SIZE 的隐藏域，通过它的值可以实现上传文件大小的软限制。

$_FILES[ ]全局数组为一个多维数组，用于获取通过 POST 方法上传文件时的相关信息。如果是单文件上传，那么数组为二维数组；如果是多文件上传，那么数组为三维数组。

$_FILES 数组中元素的含义如表 6.6 所示。



Note

表 6.6 $_FILES 数组中元素的含义

元 素 名	说 明
$_FILES[filename][name]	浏览器提供的文件名。使用价值不大，因为客户端机器上的文件名约定有可能和 Web 服务器不同（例如，如果客户机为 Windows 系统，文件名可能为“C:\PHOTOS\ME.JPG”，而服务器为 UNIX 系统，那么这个文件路径没什么意义）
$_FILES[filename][size]	已上传文件的大小单位为 B。如果用户试图上传一个过大的文件，它的大小将被置为 0
$_FILES[filename][tmp_name]	文件上传到服务器后，在服务器中的临时文件名
$_FILES[filename][type]	从客户端上传的文件类型。例如“image/gif”，主类型为“图像”，子类型为 GIF 格式的文件，“text/html”代表文本的 HTML 文件
$_FILES[filename][error]	返回在上传过程中发生错误的错误代号。错误代号有 5 种，如下所示： 0：表示没有任何错误，文件上传成功 1：表示上传文件的大小超出了 PHP 配置文件指令 upload_max_filesize 选项限制的值 2：表示上传文件的大小超出了 HTML 表单中 MAX_FILE_SIZE 选项所指定的值 3：表示文件只被上传了一部分 4：表示没有上传任何文件

PHP 中应用 is_uploaded_file()函数判断指定的文件是否是通过 HTTP POST 上传的，如果是，返回 True，则可以继续执行文件的上传操作；否则将不能够继续执行。其语法如下：

```
bool is_uploaded_file ( string filename )
```

参数 filename 必须指定类似于$_FILES['filename']['tmp_name']的变量，不可以使用从客户端上传的文件名$_FILES['filename']['name']。

通过 is_uploaded_file()函数对上传文件进行判断，可以确保恶意的用户无法欺骗脚本去访问本不能访问的文件，例如/etc/passwd。

PHP 中应用 move_uploaded_file()函数将文件上传到服务器中指定的位置。如果成功，返回 True，否则返回 False。其语法如下：

```
bool move_uploaded_file ( string filename, string destination )
```

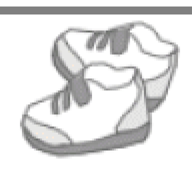
参数 filename 指定上传文件的临时文件名，即$_FILES[tmp_name]；参数 destination 指定文件上传后保存的新路径和名称。

指点迷津：

如果参数 filename 不是合法的上传文件，则不会执行任何操作，而 move_uploaded_file() 将返回 False；如果参数 filename 是合法的上传文件，但出于某些原因无法移动，同样也不会执行任何操作，而 move_uploaded_file()将返回 False，此外还会发出一条警告。

例 6.8 编写一个文件上传的实例。其步骤如下。**（实例位置：配套资源\mr\6\example\6.8）**

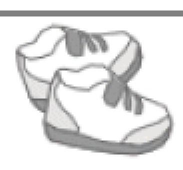
首先创建表单，设置 enctype 的属性值为 multipart/form-data，设置隐藏域对上传文件的大小进行软限制、添加文件域和提交按钮，使用 POST 方法将数据提交到当前页面。然后，通过





$_FILES 全局数组获取上传文件的数据，并且对其进行判断，通过 is_uploaded_file()函数判断指定的文件是否是通过 HTTP POST 上传的。最后应用 move_upload_file()函数将上传文件移动到指定的文件夹中。其关键代码如下：

```
<div id="one">
    <span class="three">文件上传</span>
    <form action="" method="post" enctype="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="10000000">
        <input class="one" type="file" name="text">
            <input class="two" type="submit" name="sub" value="上传"/>
    </form>
    <span class="four">
<?php
if(isset($_POST['sub'])){                    //判断是否执行提交操作
    if(!is_dir("images")){                   //判断指定文件夹是否存在
        mkdir("images");                     //创建文件夹
    }
    $file=$_FILES['text'];                   //获取表单提交的文件名称
    if($_FILES['text']['error']>0){          //判断文件是否可以上传到服务器
        echo "上传错误:";
        switch($_FILES['text']['error']){
            case 1:
                echo "上传文件大小超出配置文件规定值";
            break;
            case 2:
                echo "上传文件大小超出表单中约定值";
            break;
            case 3:
                echo "上传文件不全";
            break;
            case 4:
                echo "没有上传文件";
            break;
        }
    }else{
        if(is_uploaded_file($file['tmp_name'])){                //对提交进行验证
            $floatTime=time();
            $str=substr($file['name'],-4,4);
            $path="images/".$floatTime.$str;
            if(move_uploaded_file($file['tmp_name'],$path)){    //执行文件上传操作
                echo "上传成功，文件名称为：".$floatTime.$str;
            }
        }
    }
}
    ?>
```



```
        </span>
    </div>
```

运行结果如图 6.11 所示。





图 6.11　上传文件

6.4.10　表单验证

在使用和存储表单提交的数据时，通常需要对这些数据进行验证，验证的方法很多，例如首先在客户端使用 JavaScript，但用户可以禁用 JavaScript，甚至使用一个不支持 JavaScript 的浏览器，所以用此方法不够稳妥。

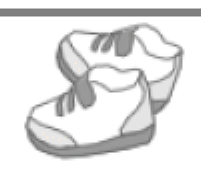
更为稳妥的方式是通过 PHP 来完成验证。验证表单元素是否为空，首先通过 isset()函数检测变量是否设置，然后通过 empty()检测变量是否为空。

例 6.9　创建一个用户登录模块，应用 isset()和 empty()函数在本页完成对用户登录信息的验证操作，其关键代码如下：**（实例位置：配套资源\mr\6\example\6.9）**

```
<form id="form1" name="form1" method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
<tr>
    <td height="30" align="center" class="STYLE1">用户名：
      <input name="user" type="text" size="16" /></td>
</tr>
<tr>
      <td height="30" align="center" class="STYLE1">密  码：
          <input name="password" type="password" size="16" /></td>
      </tr>
  <tr>
      <td height="30" align="center"><input type="submit" name="Submit" value="登录" /></td>
</tr>
</form>
<?php
if(isset($_POST['Submit'])){                    //判断登录按钮是否被设置
  $user = $_POST['user'];                       //获取用户名
  $password = $_POST['password'];               //获取密码
  if(empty($user) || empty($password)){         //验证用户名和密码是否为空
       echo "<script>alert('用户名和密码不能为空！'); window.location.href='index.php';</script>";
  }else{
       echo "输入的用户名为：$user  密码为：$password <br />";
  }
}
?>
```

运行结果如图 6.12 所示。

通过 PHP 对具体的表单元素值进行验证，如果是单纯的数字、英文字符串、字符串大小写的区分等，则 PHP 中有相应的函数可以独立完成，但如果是对电话号码、E-mail 或 IP 地址



等进行验证，则必须借助正则表达式的帮助。

例 6.10 通过 preg_match()和 preg_match_all()函数对表单中提交的手机号码和座机号码进行验证，并返回各自的匹配次数。操作步骤如下。（**实例位置：配套资源\mr\6\example\6.10**）

首先，创建 form 表单，添加表单元素，将电话号码提交到 index.php。然后，编写 PHP 脚本，通过$_POST[]方法获取表单提交的电话号码。最后，通过 preg_match()函数对座机号码进行匹配，通过 preg_match_all()函数对手机号码进行匹配。其关键代码如下：

```
<?php
    $checktel="/^(\d{3}-)(\d{8})$|^(\d{4}-)(\d{7})$|^(\d{4}-)(\d{8})$/";
                                                   //定义验证座机号码的正则表达式
    $checkphone="/^13(\\d{9})$|^15(\\d{9})$/";          //定义验证手机号码的正则表达式
    if(isset($_POST['check_tel']) && !empty($_POST['check_tel'])){     //判断是否有数据提交
        $counts=preg_match($checktel,$_POST['check_tel']);
        if($counts==1){                                         //执行验证操作
            echo "<script>alert('座机号码格式正确!');window.location.href='index.php';</script>";
        }else{
            echo "<script>alert('座机号码格式不正确!');window.location.href='index.php';</script>";
        }
    }
    if(isset($_POST['check_phone']) and !empty($_POST['Submites'])){
        $counts=preg_match_all($checkphone,$_POST['check_phone'],$arr);
        if($counts > 0){
            print_r($arr);
            echo "<script>alert('手机号码格式正确!');window.location.href='index.php';</script>";
        }else{
            print_r($arr);
            echo "<script>alert('手机号码格式不正确!');window.location.href='index.php';</script>";
        }
    }
?>
```

运行结果如图 6.13 所示。



图 6.12 验证表单元素是否为空

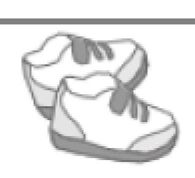


图 6.13 preg_match()和 preg_match_all()函数

上机演练

上机演练 2 设计论坛登录界面

论坛是许多网站不可或缺的模块之一，它提供一个空间让人们相互之间能够进行各种沟通







和交流。设计论坛模块的第一个步骤就是设计论坛登录界面，让用户根据自己注册的用户名和密码登录论坛。在本例中笔者设计了一个简单的论坛登录界面，效果如图 6.14 所示。



图 6.14　登录界面

上机演练 3　可以上传图片的表单

许多大型网站为了吸引用户，会为用户分配一些空间供用户使用。例如，大家经常使用的电子邮箱、QQ 个人空间等。为了能够让用户在个人空间中放置自己的内容，网站都会提供文件、图片等信息的上传功能。例如，在 QQ 个人空间中，用户可以将自己的照片上传到空间中，供好友欣赏。这里利用文件域实现图片的上传功能，效果如图 6.15 所示。

上机演练 4　以文本域的形式显示数据信息

在开发网站的过程中，一般涉及到用户注册的网站中都有一个用户注册的服务条款，个别的服务条款以整个网页进行显示，但是有些网站用户注册的服务条款的篇幅过长，就不适合使用整个页面来显示，因为这样比较浪费网络空间，最好的方式就是以文本域的形式来显示用户注册的服务条款，既节省页面，又美观大方。下面以文本域的形式显示用户注册服务条款，运行效果如图 6.16 所示。



图 6.15　可以上传图片的表单

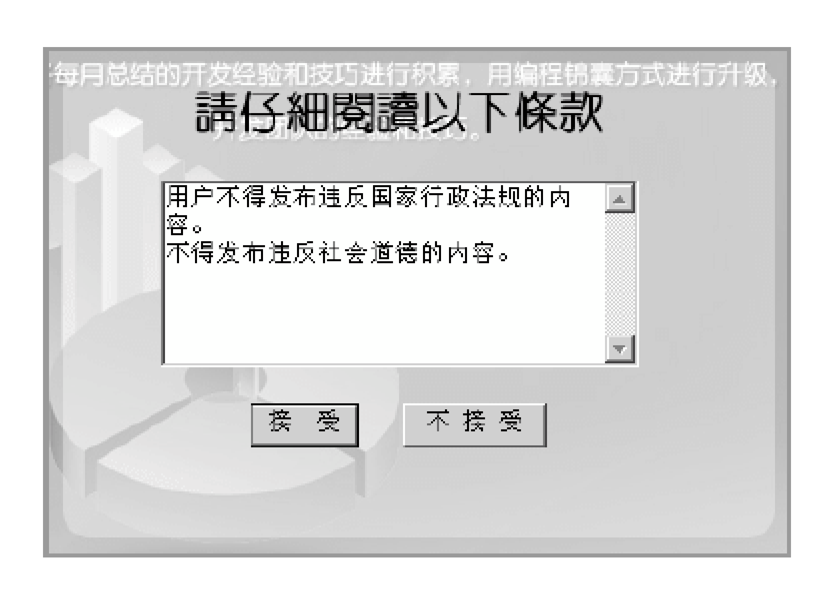


图 6.16　以文本域的形式显示注册服务条款

6.5　设置响应头

视频讲解：配套资源\mr\6\video\设置响应头.exe

服务器发送回来的 HTTP 响应包含以下信息：用于识别响应主体内容的头（header），发送响应的服务器，以及响应消息有多少字节和响应何时发出等。PHP 和 Apache 已经完成对头信息的处理：将文档识别为 HTML 和计算 HTML 页面的长度等。绝大多数 Web 程序不需要自己设置头，但如果想让服务器返回的不是 HTML，或者想设置页面的过期时间，重定向客户端浏览器到另一个地址，又或是产生一个特定的 HTTP 错误，就需要使用 header()函数来设置头。header()函数的语法如下：

```
void header ( string string [, bool replace [, int http_response_code]] )
```





header()函数的参数说明表 6.7 所示。

表 6.7 header()函数的参数说明

参　数	说　明
string	必选参数。输入的头部信息
replace	可选参数。指明是替换掉前一条类似的标头还是增加一条相同类型的标头。默认为替换，但如果将其设为 False，则可以强制发送多个同类标头
http_response_code	可选参数。强制将 HTTP 响应代码设为指定值，此参数是 php5.3.0 以后添加的

指点迷津：

设置 header 一定要在生成主体内容之前完成，这意味着所有 header()（或 setCookie()，如果想设置 Cookie）要在文件的最前面，甚至在<html>标签之前。

6.5.1　不同的内容类型

Content-Type 头指定被返回文档的类型。它通常是“text/html”，指明它是一个 HTML 文档，但还有其他一些有用的文档类型，例如“text/plain”让浏览器强制性地将内容当作纯文本来处理。该类型就类似于自动的“查看源代码”，它在调试时很有用。

例 6.11　应用 header()函数在登录页面中，以图像格式输出验证码，其步骤如下。（**实例位置：配套资源\mr\6\example\6.11**）

首先，创建 index.php 文件，编写用户登录的表单。通过 img 标签指定 png.php 文件，输出 png.php 文件中生成的验证码。其关键代码如下：

```
<img src="png.php" border="0" />
```

然后，创建 png.php 文件，通过 header()函数和 GD2 函数库中的函数生成图像验证码。其代码如下：

```
<?php
header("Content-type:image/png");              //发送头部信息，生成 png 的图片文件
$str = 'abcdefghijkmnpqrstuvwxyz1234567890';
$l = strlen($str);                             //得到字串的长度
$authnum = '';
for($i=1;$i<=4;$i++){
    $num=rand(0,$i-1);                         //每次随机抽取一位数字
    $authnum.= $str[$num];                     //将通过数字得来的字符连起来一共是4位
}
srand((double)microtime()*1000000);
$im = imagecreate(50,20);                      //图片宽与高
$black = imagecolorallocate($im, 0,0,0);
$white = imagecolorallocate($im, 255,255,255);
$gray = imagecolorallocate($im, 200,200,200);
imagefill($im,68,30,$black);                   //将4位整数验证码绘入图片
imagestring($im, 5, 8, 2, $authnum, $white);   //字符在图片的位置
imagepng($im);
imagedestroy($im);
?>
```

运行结果如图 6.17 所示。





图 6.17　图片验证码

6.5.2　重定向

通过 header()函数可以向浏览器发送一个新的 URL，并让浏览器转向到该地址。这样的重定向（redirection）操作，只需要通过设置 Location 头即可。例如，通过 header()函数重定向到 http://www.mrbccd.com，其代码如下：

```
header("Location: http://www.mrbccd.com");
```

重定向操作更倾向于绝对路径，如果提供相对的 URL（如/index.php），重定向会在服务器内部进行。这种方法很少用，因为浏览器并不知道它得到的页面是否是所请求的。如果在新的文档中存在相对 URL，浏览器会将它们解释成相对于所请求的文档，而不是被发送的文档。

6.5.3　设置过期时间

服务器可以显式地通知浏览器（或者那些存在于服务器的浏览器之间的代理服务器缓存）文档的过期时间。代理服务器和浏览器缓存在过期之前可保持文件或提前结束它。重新载入一个被缓存的页面不需要和服务器进行通信，但是尝试获取一个已经过期的文档就需要与服务器取得联系。

为一个文档设置过期时间，应用的是 Expires 头：

```
header("Expires:     Mon,     08     Jul     2011     08:08:08     GMT");
```

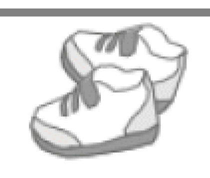
例如：控制文档在页面生成两小时后过期。使用 time()和 gmstrftime()函数生成过期日期字符串：

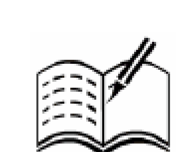
```
<?php
$now=time();                    //获取系统当前时间戳
$then=gmstrftime("%a,%d %b %Y %H:%M:%s GMT",$now + 60*60*2);         //格式化时间
header("Expires:$then");              //定义文档过期时间
?>
```

6.5.4　HTTP 认证

HTTP 认证（HTTP authentication）通过请求的 header 和响应状态来工作。浏览器可以将用户名和密码放在请求的头中发送。如果认证凭证（credential，即指用户名和密码）未发送或不匹配，服务器将发送一个“401 Unauthorized”响应并通过 WWW 认证头来确定当前认证的区域（realm）（一个字符串，诸如“Mary's Pictures”或“Your Shopping Cart”）。这通常会导致浏览器弹出一个 Enter username and password for…对话框，且该页面会重新请求更新头中的认证凭证。

为了用 PHP 来处理认证，可检查用户名和密码（$_SERVER 数组中的 PHP_AUTE_USER 和 PHP_AUTH_PW 两个元素）并调用 header()函数来设置区域，然后发送一个“401 Unauthorized”





Note

响应。其关键代码如下：

```
header('WWW-Authenticate:Basic realm="Top Secret Files"');
header("HTTP/1.0 401 Unauthorized");
```

header()函数还可以强制客户端每次访问页面时获取最新资料，而不是使用存在于客户端的缓存。其关键代码如下：

```
header("Expires: Mon, 08 Jul 2011 08:08:08 GMT"); //设置页面的过期时间用格林威治时间表示
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . "GMT");        //设置最后更新日期用格林威治时间表示，使浏览器获取最新资料
header("Cache-Control: no-store,no-cache, must-revalidate");    //控制页面不使用缓存
header("Cache-Control: post-check=0,pre-check=0",false);        //控制页面不使用缓存
header("Pragma: no-cache");                 //参数（与以前的服务器兼容），即兼容HTTP1.0协议
```

6.5.5　文件下载

根据服务器返回的头信息，应用 header 函数可以完成文件的下载操作。其关键代码如下：

```
header("Content-type:application/octet-stream");          //输出MIME类型
header("Accept-ranges:bytes");                             //接受的范围单位
header("Accept-length:".filesize($path));                  //文件长度
header("Content-Disposition:attachment;filename=".$filename); //默认时文件保存对话框中的文件名称
```

例如：通过 header()函数生成 Excel 格式的下载文件，其关键代码如下：

```
<?php
header("Content-type:application/vnd.ms=excel");           //设置 HTTP 标头
header("Content-Disposition:filename=会议报表.xls");        //定义下载文件名称
?>
```

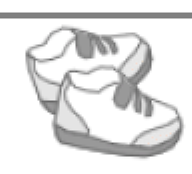
上机演练

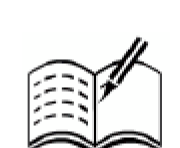
上机演练 5　验证用户注册信息是否合理

判断用户输入的注册信息是否合理，当用户输入的注册信息不合理时给出相应的提示，运行结果如图 6.18 所示。



图 6.18　验证用户注册信息是否合理





上机演练 6　省市级联动菜单

在进行网站开发的过程中，经常会用到相互关联的菜单，例如省市县级联动或省市级联动菜单。下面创建省市级联动菜单，其运行效果如图 6.19 所示。

上机演练 7　省市县级联动菜单

既然可以创建省市级联动菜单，那么同样可以创建省市县级联动菜单。运行效果如图 6.20 所示。





图 6.19　省市级联动菜单



图 6.20　省市县级联动菜单

上机演练 8　实现复选框中的全选、反选和不选

在网站的后台管理系统中，经常需要执行一些批量删除或修改的操作，而此时就需要对指定的数据进行选中，为了简洁、方便地完成选中的操作，最简单、有效的方法就是实现复选框的全选、反选和不选功能。下面通过 JavaScript 脚本实现复选框的全选、反选和不选功能，如图 6.21 所示为全选时的状态。



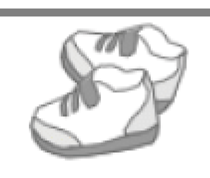
图 6.21　实现全选

上机演练 9　上传图片预览

在上传图片时最好为程序设置预览功能，以防止用户意外传错图片。同时还要有可以获取图片的地址，以方便用户网站的推广。通过表单元素与 JS 脚本的结合应用，完成图片预览和图片地址获取的功能，运行结果如图 6.22 所示。

上机演练 10　通过下拉列表选择头像

在腾讯的 QQ 系统中，我们可以通过单击头像而实现更换新的头像。其实该功能相对来说



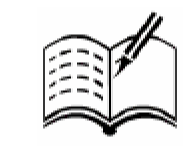
是比较简单的。下面通过下拉列表实现头像的选择，如图 6.23 所示。



图 6.22　上传预览图片



图 6.23　选择头像



6.6　维 持 状 态

视频讲解：配套资源\mr\6\video\维持状态.exe

HTTP 是一个无状态的通信协议，这意味着一旦 Web 服务器完成客户端的 Web 页面请求，它们之间的连接也就断开了。换句话说，我们无法使服务器识别来自于同一个客户端的一系列请求。

但是状态是很有用的。例如，如果不能跟踪来自于同一个用户的一系列请求，就无法设计一个购物车程序。你需要知道用户在什么时候购物车中放入物品，什么时候删除，结账时购物车中有些什么物品？

为了解决 Web 状态不维持的问题，程序员提出了以下几种在两次请求间跟踪状态信息的方法（这称为会话跟踪，session tracking）。

方法一：使用隐藏的表单字段来传递信息。因为 PHP 以对待正常表单字段的方式来对待隐藏表单字段，所以可以通过$_GET 和$_POST 数组访问隐藏表单字段的值。这样，利用隐藏表单字段就可以传递购物车的全部信息。

方法二：为每一个访问的用户分配一个唯一标识符，并且通过一个隐藏表单字段传送 ID。隐藏表单字段可以在所有的浏览器中使用，但它们只能用于一系列动态创建的表单，所以它们不如其他的技术通用。

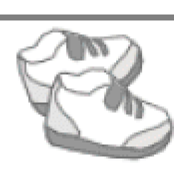
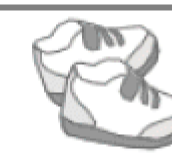
指点迷津：

方法二不太好用，因为表单提交的下一个页面也必须含有表单，在多个页面间传递信息，则多个页面都需要表单，实现起来比较繁琐。实际应用中常用于“向导型”页面，如注册程序，通过“下一步”按钮，用户跳到下个页面继续注册，可在下个页面保持上一页中用户输入的信息。

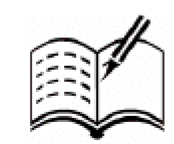
方法三：重写 URL，用户访问的每一个 URL 都加上附加的信息，附加信息作为 URL 的参数。例如，如果为每个用户分配一个唯一的 ID，则可以在每个 URL 中包含该 ID。

```
<a href="index.php?id=1">重写 URL </a>
```

如果要动态修改所有的本地链接以包含一个用户 ID，那么可以在程序中跟踪单个用户的信息。URL 重写可用于所有动态生成的文档，而不仅是包含表单的文档，但是很显然为每个页面重写是非常乏味的。

方法四：使用 Cookie 实现状态维持。Cookie 是服务器给客户端的一些信息。在后继的请求中，客户端将把信息返回给服务器，这样就可以标识自己的身份。Cookie 对于浏览器重复访问时保持客户端信息很有用，但它也有不足的地方，主要表现在有些浏览器不支持 Cookie，即使支持，用户也可以禁用 Cookie，所以使用 Cookie 来维持状态的程序需要使用另一个叫回退（fallback）的机制。




方法五：在 PHP 中维持状态的最好方法是使用其内置的会话跟踪系统（session-tracking system）。该系统允许创建一个持久性变量。从程序的不同页面和同一用户对站点的不同访问都可以访问该变量。PHP 的会话跟踪机制使用 Cookie（或 URL）在后面优雅地解决了维持状态的大部分问题，并为用户考虑了所有细节。

6.6.1 Cookie

Cookie 是一个包含多个字段的字符串。一个服务器可发送包含在响应头中的一个或多个 Cookie 给浏览器。Cookie 中的一些字段指明哪些需要浏览器将 Cookie 发送至页面。Value 字段是 Cookie 中的有效内容——服务器能够将一些它所需要的数据保存在那里（有限制），诸如用于标识用户的唯一 ID、首选项（preference）等。

在 PHP 中使用 setCookie()函数向浏览器发送 Cookie。setCookie()函数的语法如下：

```
bool setCookie(string name[,string value[,int expire[, string path[,string domain[,int secure]]]]])
```

setCookie()函数的参数说明如表 6.8 所示。

表 6.8　setCookie()函数的参数说明

参　　数	说　　明	举　　例
name	Cookie 的变量名	可以通过$_COOKIE['Cookiename']调用变量名为 Cookiename 的 Cookie
value	Cookie 变量的值，该值保存在客户端，不能用来保存敏感数据，建议大小不要超过 3.5KB	可以通过$_COOKIE['values']获取名为 values 的值
expire	Cookie 的过期时间，如果没有指定过期时间，则浏览器将把此 Cookie 保存在内存中而不是硬盘中，当浏览器退出页面访问时，Cookie 也就消失了。过期时间是以格林威治标准时间 1970 年 1 月 1 日为开始时间的，单位是秒	如果不设置 Cookie 的过期时间，那么 Cookie 将永远有效，除非手动将其删除
path	Cookie 在服务器端的有效路径	如果在/store/front/cart.php 文件中设置了 Cookie，且没有指定路径，那么 Cookie 将被浏览器返回给服务器上所有/store/front 目录下的页面
domain	Cookie 有效的域名	如果要使 Cookie 在 mrbccd.cn 域名下的所有子域都有效，应将其设置为 mrbccd.cn
secure	浏览器只通过 HTTPS 连接上传 Cookie。它的默认值为 False，其含义是允许通过不可靠连接（即普通的 HTTP 连接，而非 HTTPS 安全连接）发送 Cookie	如果值为 1，则 Cookie 只能在 HTTPS 连接上有效；如果值为默认值 0，则 Cookie 在 HTTP 和 HTTPS 连接上均有效

指点迷津：

因为 Cookie 是作为头发送的，所以必须在文件体被传递之前调用 setCookie()函数。





Cookie 是一种在远程客户端存储数据并以此来跟踪和识别用户的机制。简单地说，Cookie 是 Web 服务器暂时存储在用户硬盘上的一个文本文件，并随后被 Web 浏览器读取。当用户再次访问 Web 网站时，网站通过读取 Cookie 文件记录这位访客的特定信息（如上次访问的位置、花费的时间、用户名和密码等），从而迅速作出响应，例如，在页面中不需要输入用户的 ID 和密码即可直接登录网站。



Cookie 文本文件以（用户名@网站地址[数字].txt）格式存储于客户端硬盘的指定文件夹下。例如，客户端机器为 Windows 2000/XP/2003 操作系统，系统盘为 C 盘，当通过 IE 浏览器访问 Web 网站时，Web 服务器会自动以（用户名@网站地址[数字].txt）格式生成 Cookie 文本文件，并存储在用户硬盘的指定位置，如图 6.24 所示。

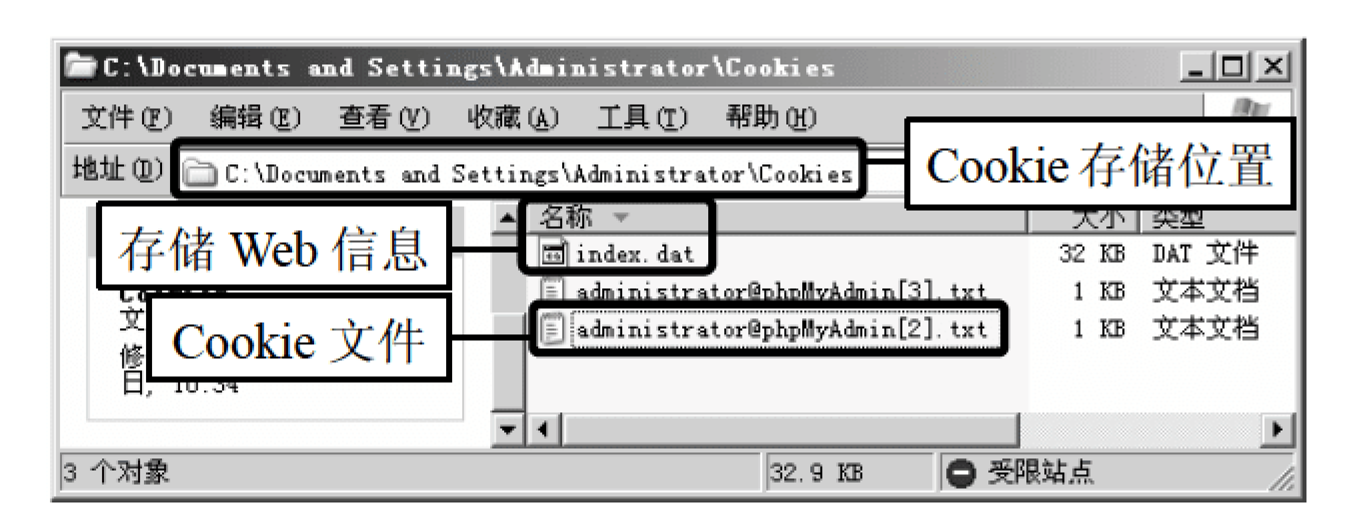


图 6.24　Cookie 文件的存储路径

多学两招：

在 Cookies 文件夹下，每个 Cookie 文件都是普通的文本文件，而不是程序。文本文件中的内容大多都经过了加密处理，因此，表面看来只是一些字母和数字的组合，而只有服务器的 CGI 处理程序才知道它们真正的含义。

当浏览器发送一个 Cookie 给服务器时，可以用$_COOKIE 数组来访问。键名为 Cookie 的名称，值为 Cookie 的 value 字段。例如：

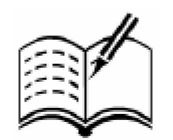
```
<?php
$text=$_COOKIE['text']
setCookie('text',$text);                    //设置Cookie变量
?>
```

例 6.12　通过 Cookie 保存用户登录信息，其步骤如下。**（实例位置：配套资源\mr\6\example\6.12）**

相信大家对开心网都很熟悉，它的主页登录界面就采用了 Cookie 保存用户登录信息的机制，将用户名和密码根据用户指定的时间长期保存在客户端的 PC 上。其功能实现的步骤如下：

（1）创建 index.php 文件，首先判断 COOKIE 变量是否存在，如果存在则直接将其赋给指定的变量，否则，获取表单提交的用户名、密码和数据保存时间，通过 setcookie()函数创建 COOKIE。其关键代码如下：

```
<?php
header("Content-Type:text/html;charset=utf-8");              //定义编码格式
if(isset($_COOKIE['text'])){                                  //判断COOKIE是否存在
$text=$_COOKIE['text'];                                       //如果存在则将其赋给指定的变量
$pwd=$_COOKIE['pwd'];
}else{
setcookie('text',$_POST['text'],time()+$_POST['radio']);      //否则设置Cookie变量
setcookie('pwd',$_POST['pwd'],time()+$_POST['radio']);
```



```
$text="";                                             //设置变量初始值为空
$pwd="";
}
?>
```

（2）创建表单，提交用户名、密码和保存时间。其代码如下：

```
<form action=""method="post">
    用户名：<input type="text"class="on" name="text" value="<?php echo $text;?>"><br>
    密  码：<input class="on" type="password" name="pwd" value="<?php echo
$pwd;?>"><br>
    <input type="radio" name="radio" value="3600" checked>1小时
    <input type="radio" name="radio" value="86400">1天
    <input type="radio" name="radio" value="604800">1星期
    <input class="o" type="submit" name="sub" value="登录">  
    <input class="o" type="reset" name="res" value="重置">
</form>
```

（3）判断提交按钮的值是否设置，提醒用户名和密码不能为空。

```
<?php                                                 //启动SESSION
if(isset($_POST['sub'])){                             //当页面中存在变量sub时
    if($_POST['text']=="" and $_POST['pwd']==""){     //判断并输出提示
        echo "<script>alert('文本框内容为空');</script>";
    }
}
?>
```

运行效果如图 6.25 所示。

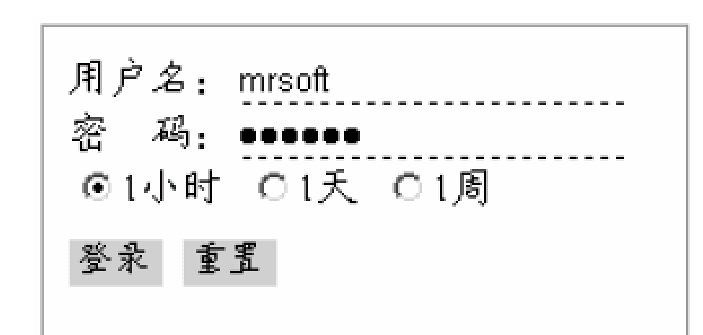


图 6.25　通过 Cookie 保存用户登录信息

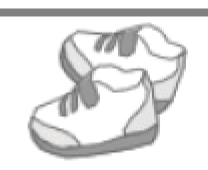
脚下留神：

不是所有的客户端都支持和接受 Cookie，即使支持，用户也可能禁用 Cookie 功能。另外，Cookie 规范规定了 Cookie 不能大于 4KB，一个域只能发送 20 个 Cookie，而且一个客户端只能存储 300 个 Cookie，虽然有些浏览器可能限制宽松一些，但不能依赖于此。最后，无法控制客户端浏览器使 Cookie 过期——当浏览器有能力并且需要新增一个 Cookie 时，它将抛弃一个没有过期的 Cookie。在使 Cookie 短时间内过期时要小心，过期时间取决于客户端机器的时钟，它未必和服务器时钟相同，所以不能依赖于快速过期。

虽然有诸多的限制，但是 Cookie 仍不失为一种在浏览器重复访问时保持信息的实用方法。

6.6.2　会话

PHP 内置了对 Session 的支持，以帮助用户处理所有的 Cookie 操作，并提供不同页面和多次访问都可用的持久性变量。Session 可以使用户十分容易地创建多页面的表单（如购物车），





在页面到页面之间保存用户鉴别信息以及保存永久用户在某一站点上的首选项。

每个第一次访问的用户都会被分配一个唯一的与别人不同的 session ID。默认地，session ID 存放在一个名为 PHPSESSION 的 Cookie 中。如果用户的浏览器不支持 Cookie 或者禁用了 Cookie，则 session ID 会被传递到 Web 站点的 URL 内。

每个 session 都有一个关联的数据存储，可以在页面开始时注册来自于数据存储的变量，并在页面结束时将其保存回数据存储。注册的变量在多个页面之间都可用，且一个页面内对变量的改动在其他页面也是可见的（也就是说变量的作用域跨越了多个页面）。

创建一个会话需通过以下 4 个步骤：启动会话、注册会话、使用会话和删除会话。

1. 启动会话

启动 PHP 会话使用 session_start()函数，其语法如下：

```
bool    session_start(void)
```

该函数的参数为空。

脚下留神：

在使用 session_start()函数之前浏览器不能有任何输出，否则会产生类似于如图 6.26 所示的错误，即使在 PHP 脚本之外存在空格也不可以。

```
在session_start()函数之前输出的数据 Warning: session_start
(): Cannot send session cache limiter - headers already
sent (output started at F:\PkhPHP\www\MR\Instance\10
\10.1\index.php:2) in F:\PkhPHP\www\MR\Instance\10\10.1
\index.php on line 4
```

图 6.26　在使用 session_start()函数前输出字符串产生的错误

2. 注册会话

会话变量被启动后，全部保存在数组$_SESSION 中。通过数组$_SESSION 创建一个会话变量很容易，只要直接为该数组添加一个元素即可。

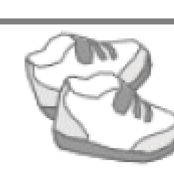
例如：启动会话，创建一个 SESSION 变量并赋予空值，代码如下：

```
<?php
session_start();                  //启动Session
$_SESSION["user"] = null;         //声明一个名为user的变量，并赋空值
?>
```

3. 使用会话

例如：判断页面中是否存在会话 ID，如果不存在，就创建一个，并且使其能够通过全局数组$_SESSION 进行访问；如果已存在，则将这个已注册的会话变量载入以供用户使用。代码如下：

```
<?php
session_start();
if(isset(session_id())){              //判断SESSION是否设置
    echo $_SESSION['mr'];             //如果设置直接输出SESSION的值
}else{
    $_SESSION['mr']="mrsoft";1        //否则为session赋值
}
?>
```







4. 删除会话

删除会话有 3 种方式：删除单个会话、删除多个会话和结束当前会话。

☑ 删除单个会话

删除会话变量，与删除数组中的指定元素相同，通过 unset()函数完成。unset()函数的语法如下：

```
unset ( $_SESSION['user'] ) ;
```

其参数是$_SESSION 数组中的指定元素，该参数不可以省略。通过 unset()函数一次删除数组中的一个元素；如果通过 unset()函数一次注销整个数组（unset($_SESSION)），那么会禁止整个会话功能，而且无法将其恢复，用户也不能再注册$_SESSION 变量。所以，如果读者要删除多个或全部会话，可以采用下面的两种方式。

☑ 删除多个会话

如果要一次注销所有的会话变量，可以将一个空的数组赋值给$_SESSION，代码如下：

```
$_SESSION = array() ;
```

☑ 结束当前会话

如果整个会话已经结束，那么就可以使用 session_destroy()函数结束当前的会话。session_destroy()函数的语法如下：

```
session_destroy();
```

该函数清空会话中的所有资源，彻底销毁 SESSION。

5. 自定义存储

PHP 默认把 session 信息存储在服务器的临时文件夹中，每个 session 变量被存在不同的文件中，每个变量都被序列化成适当的格式。用户可以通过编辑 php.ini 文件来改变设置，也可以通过 session_save_path()函数重新配置目前 SESSION 的存储位置。

通过改变 php.ini 中 session.save_path 选项的值来改变 session 文件的存位置。如果用户在一个共享的服务器上使用自己安装的 PHP，则可以把存放目录改为用户有权限的目录，这样服务器上的其他用户就无法访问你的 session 文件。

通过 session_save_path()函数取得或者重新配置目前 SESSION 的存放路径。其语法如下：

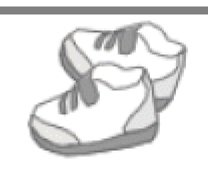
```
string session_save_path([string path])
```

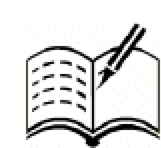
如果设置参数 path，表示重新设置 SESSION 的存储路径；如果不设置参数 path，表示直接获取当前 SESSION 的存储路径。

例如：将 session 文件保存在当前目录的 session_files 下。代码如下：

```
<?php
$path = './session_files/';                //设置session存储路径
session_save_path($path);                  //重新设置session的存储路径
session_start();                           //初始化session
$_SESSION['id'] = "明日科技";
?>
```

在当前的 session 存储中，PHP 能以两种方式存储 session 信息——PHP 内置格式和 WDDX（http://www.openwddx.org/）。可以通过设置 php.ini 中的 session.serialize_handler 的值来改变存储格式。其中 php 表示默认格式，wddx 表示 WDDX 格式。





6. SESSION 数据库存储

将 SESSION 存储于指定文件中，虽然不至于将临时文件夹填满而造成站点瘫痪，但是如果网站访问量超大，仍然会影响运行速度。因此可将 SESSION_ID 存储于数据库中，那么查询的速度受到的影响就会小很多。

将 SESSION_ID 存储于数据库中应用的是 session_set_save_handler()函数。它调用自定义函数，完成将 SESSION_ID 存储于数据库中的操作。其语法如下：

```
bool session_set_save_handler ( string open, string close, string read, string write, string destroy, string gc)
```

session_set_save_handler()函数的参数说明如表 6.9 所示。

表 6.9　session_set_save_handler()函数的参数说明

参　数	说　明
open(save_path,session_name)	找到 session 存储地址，取出变量名称
close()	不需要参数，关闭数据库
read(key)	读取 session 键值，key 对应 session_id
write(key,data)	其中 data 对应设置的 session 变量
destroy(key)	注销 session 对应 session 键值
gc(expiry_time)	清除过期 session 记录

这是一个非常特殊的函数，因为一般函数的参数都是变量，但是此函数的参数为 6 个函数。通过这 6 个函数可完成 SESSION 的数据库存储。

（1）封装 session_open()函数，将数据库连接，代码如下：

```
function _session_open($save_path,$session_name){
    global $handle;
    $handle = mysql_connect('localhost','root','root') or die('数据库连接失败');//连接MySQL数据库
    mysql_select_db('db_database11',$handle) or die('数据库中没有此库名');    //找到数据库
    return(true);
}
```

指点迷津：

我们看到$save_path 和$session_name 两个参数并没有用到，在这里可以将其去掉，但是希望读者可以加入，因为一般使用都是存在这两个变量的。我们应养成一个好的习惯。

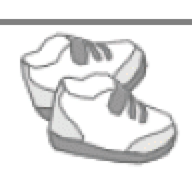
（2）封装 session_close()函数，关闭数据库连接，代码如下：

```
function _session_close(){
    global $handle;
    mysql_close($handle);
    return(true);
}
```

指点迷津：

在上述参数中不需要任何参数，所以不论是 Session 存储到数据库中，还是存储到文件中只需返回 True 就可以。但是如果是 MySQL 数据库最好将数据库关闭，这样保证以后不会出现麻烦。

（3）封装 session_read()函数，在函数中我们设定当前时间的 UNIX 时间戳，根据$key 值



Note

查找 Session 名称及内容，代码如下：

```
function _session_read($key){
    global $handle;                    //全局变量$handle连接数据库
    $time = time();                    //设定当前时间
    $sql = "select session_data from tb_session where session_key = '$key' and session_time > $time";
    $result = mysql_query($sql,$handle);
    $row = mysql_fetch_array($result);
    if ($row){
        return($row['session_data']);          //返回Session名称及内容
    }else{
        return(false);
    }
}
```

指点迷津：

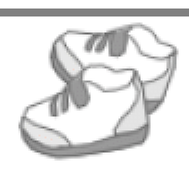
我们存储进数据库中的 session_expiry 是 UNIX 时间戳。

（4）封装 session_write()函数，函数中设定 Session 失效时间，查找到 Session 名称及内容，如果查询结果为空，则将页面中 Session 根据 session_id、session_name、失效时间插入数据库中；如果查询结果不为空，则根据$key 修改数据库中 Session 存储信息，返回执行结果，代码如下：

```
function _session_write($key,$data){
    global $handle;
    $time = 60*60;                                  //设置失效时间
    $lapse_time = time() + $time;                   //得到UNIX时间戳
    $sql = "select session_data from tb_session where session_key = '$key' and session_time > $lapse_
time";
    $result = mysql_query($sql,$handle);
    if (mysql_num_rows($result) == 0 ) {            //没有结果
        $sql = "insert into tb_session values('$key','$data',$lapse_time)";    //插入数据库sql语句
        $result = mysql_query($sql,$handle);
    }else{
        $sql = "update tb_session set session_key = '$key',session_data = '$data',session_time =
$lapse_ time where session_key = '$key'";           //修改数据库sql语句
        $result = mysql_query($sql,$handle);
    }
    return($result);
}
```

（5）封装 session_destroy()函数，根据$key 值将数据库中的 Session 删除，代码如下：

```
function _session_destroy($key){
    global $handle;
    $sql = "delete from tb_session where session_key = '$key'";      //删除数据库sql语句
    $result = mysql_query($sql,$handle);
    return($result);
}
```





（6）封装 session_gc()函数，根据给出的失效时间删除过期的 Session，代码如下：

```
function _session_gc($expiry_time)
{
    global $handle;
    $lapse_time = time();                    //将参数$expiry_time赋值为当前时间戳
    $sql = "delete from tb_session where expiry_time < $lapse_time";      //删除数据库sql语句
    $result = mysql_query($sql,$handle);
    return($result);
}
```

例 6.13　通过 session_set_save_handler()函数实现 Session 数据库存储，其步骤如下。**（实例位置：配套资源\mr\6\example\6.13）**

（1）封装 session_set_save_handler()函数的 6 个参数。

（2）调用 session_set_save_handler()函数，初始化 SESSION 变量，同时定义 SESSION 变量存储的数据。其关键代码如下：

```
    session_set_save_handler('_session_open','_session_close','_session_read','_session_write','_session_d
estroy','_session_gc');
    session_start();
    //下面为我们定义的Session
    $_SESSION['user'] = 'mr';
    $_SESSION['pwd'] = 'mrsoft';
```

现在看一下数据库中表 session 的内容，如图 6.27 所示。

←T→	session_key	session_data	session_time
☐ ✎ ✕	f5e66a38742eb37743f20d353f71e0c4	user|s:2:"mr";pwd|s:6:"mrsoft";SID|b:1;	1260846901

图 6.27　Session 数据库存储

上机演练

上机演练 11　SESSION 更换聊天室界面

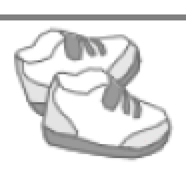
SESSION 可以实现数据在页面之间的传递，并且在 SESSION 的生命周期中一直有效。下面运用 SESSION 的这个特性，编写一个简单的聊天室换肤功能。在聊天室中，根据提交的颜色值更换聊天室的背景颜色，其运行结果如图 6.28 所示。

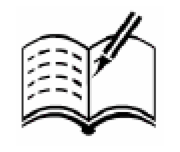


图 6.28　更换聊天室背景颜色

上机演练 12　掌控登录用户的权限

在论坛中，如果是管理员则可以发布公告信息，如果是普通用户，那么就不可以发布公告







信息。那么程序中是如何对登录用户的权限进行判断的呢？答案是通过 SESSION 变量掌控登录用户的权限。如果是管理员登录，那么可以添加公告信息，而如果是普通用户则只可以浏览论坛中的帖子，其运行结果如图 6.29 和图 6.30 所示。



图 6.29 管理员登录

图 6.30 发布留言

上述以管理员身份进行登录，进入到 main.php 主页后，可以单击“留言”超链接，发布留言信息；如果是以普通用户身份登录，那么在 main.php 主页中，“留言”超链接是不可以单击的，因为普通用户不具备发布留言的权限。

上机演练 13 屏蔽页面刷新对计数器的影响

计数器用来统计一个网站被访问次数，它体现一个网站的受关注程度。下面编写一个网站计数器程序，并且应用 SESSION 屏蔽刷新页面对计数器的影响。当用户访问时计数器的值会增加一次，但是无论如何刷新页面，计数器的值都不会再次增加，只有重新打开页面时计数器的值才会发生变化，其运行结果如图 6.31 所示。

图 6.31 SESSION 变量屏蔽页面刷新对计数器的影响

上机演练 14 SESSION 购物车

购物车是在网上购物时使用的一个临时存储商品的“车辆”，它为用户在网上购物提供很大的方便，不用担心一次购买多个商品时而要进行多次提交结算的操作，都可以将其放入购物车中，等选购完所有商品后一起进行结算。通过 SESSION 技术创建一个购物车，其运行结果如图 6.32 所示。



图 6.32 SESSION 购物车





本 章 摘 要



1．HTTP 协议的基本原理。



2．PHP 中获取服务器信息的方法。
3．表单处理技术，包括表单的创建、表单元素的添加以及获取表单中提交的数据。
4．header()函数获取服务器返回的信息。
5．Cookie 技术。
6．会话技术。
7．在上机演练和实战模拟栏目中还融入了 JavaScript 脚本技术。

习　　题

1．在 JavaScript 脚本中，哪种写法表示文本框（name）中的值？（　　）

A．$_POST['name']　　B．$_GET['name']
C．var name　　D．form1.name.value

2．在 JavaScript 脚本中，弹出对话框，需使用 window 对象的什么方法？（　　）

A．echo　　B．window.location.href
C．alert　　D．print

3．设置表单提交跳转页，需通过（　　）属性？

A．action　　B．method
C．name　　D．value

4．对超链接传递的数据进行编码，使用什么函数？（　　）

A．action　　B．alert
C．urldecode()　　D．urlencode()

5．通过超链接传递数据时，设置单击超链接时打开新的窗口，使用 target 属性的哪个内置窗口名称？（　　）

A．_self　　B．_blank　　C．_parent　　D．_top

6．SESSION 机制是（　　）。

A．SESSION 是客户端机制　　B．SESSION 是服务器端机制
C．SESSION 是网页机制

7．Cookie 机制是（　　）。

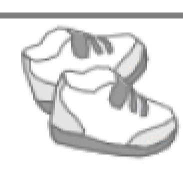
A．Cookie 是客户端机制　　B．Cookie 是服务器端机制
C．Cookie 是网页机制

8．下面文件中哪个是启动 SESSION 语句的？（　　）

A．session_start()　　B．SESSION_START()　　C．setCookie()

9．判断某个 SESSION 或 Cookie 变量使用哪个函数？（　　）

A．is_dir()　　B．is_int()　　C．isset()



Note

10．看下面代码：

```
session_start();
$_SESSION['mr']='明日科技';
```

输出$_SESSION['mr']变量的值为（　　）。

A．mr　　B．MR　　C．明日科技

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 4 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　日期选择器

基于目前日期型数据格式有多种，采用录入方式相对来说比较烦琐，而且采用这种方式也不利于日期格式的统一，所以可以在信息录入页面中加入一个简单的日期选择器来解决上述问题。在新奥家电连锁后台管理系统的销售查询页面中，单击“售货日期”文本框后的日期选择图标，会弹出“日期选择器”对话框，选择售货日期的起始日期，单击“确定”按扭，即可成功地将选择的日期添加到对应的日期文本框中，运行结果如图 6.33 所示。

实战模拟 2　树形导航菜单

树形结构能够以层次形式展示信息，用它来描述具有上下级关系的内容再恰当不过。利用菜单的树形结构来描述企业人事组织架构，效果如图 6.34 所示。



图 6.33　日期选择器

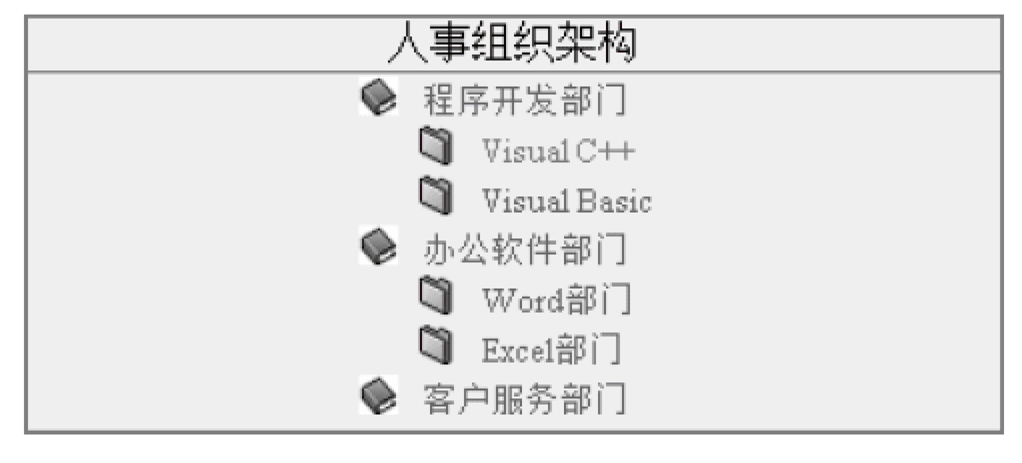
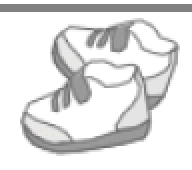


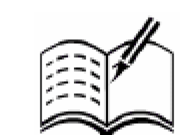
图 6.34　树形导航菜单

实战模拟 3　单点登录

现在越来越多的企业、个人喜欢用互联网来进行沟通和交流，诸如留言板、聊天室、论坛和博客等交流和展示自己的方法层出不穷。而且现在出现一个企业或者个人就拥有上述多种交流的方式，因为上述每种交流方式都具有自己的身份验证机制，这样势必会造成：如果某位用户要以会员的身份访问网站，则需要在每个网站上注册，并且通过身份验证后，才能以会员的身份访问网站；即使用户以同样的用户名与密码在每个网站上注册时，虽然可以在避免用户名与密码的忘记和混淆方面有一定的作用，但是如果用户在某一时间段内，访问网站的多个页面或者在不同网站间跳转时，仍然需要用户登录后才能以会员的身份访问网站。这样不仅给用户带来了不便，而且在登录时也浪费了服务器的空间。

如果所有的网站都能够实现单点登录，则不仅在用户体验方面有所提高，而且真正体现了





集团多个网站的兄弟性。通过这种有机结合，不但便于对其进行管理，而且能更好地体现公司大平台、大渠道的理念。同时，这样做也利于网站的相互促进与相互宣传。

运行单点登录程序，如果是第一次登录，那么将进入到登录页面，登录成功后将进入到博客主页，如图 6.35 所示。另外，还可以单击博客主页中的“我的论坛”超链接，以此用户的身份直接进入到论坛系统主页中，如图 6.36 所示。同样也可以单击论坛系统中的“博客”跳转到博客的页面中。



图 6.35　进入博客主页面



图 6.36　进入论坛系统主页



MySQL 数据库

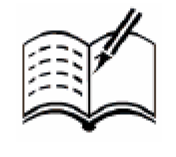
（自学视频、源程序：配套资源\mr\7\）

数据库作为程序中数据的主要载体，在整个项目中扮演着重要的角色。PHP自身可以与大多数数据库进行连接，但MySQL数据库是开源界所公认的与PHP结合得最好的数据库，且具有安全、跨平台、体积小和高效等特点，可谓是PHP的“黄金搭档”。本章将对MySQL数据库的基础知识进行系统讲解，为读者在第8章中实现PHP与MySQL数据库的完美结合打下坚实的基础。

学习摘要：

- MySQL 概述
- MySQL 服务器的启动和关闭
- 操作 MySQL 数据库
- 操作 MySQL 数据表
- 操作 MySQL 数据
- MySQL 数据类型
- phpMyAdmin 管理 MySQL 数据库





7.1　MySQL 概述

学习编程语言，至少要掌握一种数据库，而对于学习 PHP 语言，掌握 MySQL 显得更加重要。虽然现在 PHP 对数据库的支持越来越多，如 Access、MS SQL Server、Oracle、DB2 等，但是在 LAMP 的开发模式中，MySQL 仍然牢牢占据一席之地。

7.1.1　MySQL 的特点

MySQL 具有以下特点：

- ☑ MySQL 是一个关系数据库管理系统，把数据存储在表格中，使用标准的结构化查询语言——SQL 访问数据库。
- ☑ MySQL 是完全免费的，在网上可以任意下载，同时可以查看到它的源文件，并且可以进行必要的修改。
- ☑ MySQL 服务器的功能齐全，运行速度极快，十分可靠，具有很好的安全性。
- ☑ MySQL 服务器在客户、服务器或嵌入系统中使用，是一个客户机/服务器系统，能够支持多线程以及多个不同的客户程序和管理工具。

7.1.2　SQL 和 MySQL

SQL（Structured Query Language，结构化查询语言），与其说是一门语言，倒不如说是一种标准——数据库系统的工业标准。大多数 RDBMS 开发商的 SQL 都基于该标准，虽然在有些地方并不是完全相同，但这并不妨碍对 SQL 的学习和使用。

下面给出 SQL 标准的关键字及其功能，如表 7.1 所示。

表 7.1　SQL 标准的关键及其功能

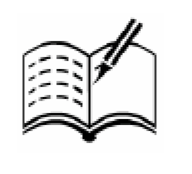
功 能 类 型	SQL 关键字	功　能
数据查询语言	Select	从一个或多个表中查询数据
数据定义语言	Create/Alter/Drop table Create/Alter/Drop index	创建/修改/删除表 创建/修改/删除索引
数据操纵语言	Insert Delete Update	向表中插入新数据 删除表中的数据 更新表中现有的数据
数据控制语言	Grant Revoke	为用户赋予特权 收回用户的特权

在 MySQL 中，不仅支持 SQL 标准，而且还对其进行了扩展，使得它能够支持更为强大的功能。下面介绍 MySQL 支持的 SQL 关键字，如表 7.2 所示。

表 7.2　MySQL 支持的 SQL 关键字

SQL 关键字	功　能
创建、删除和选择数据库	Create/Drop database/Use
创建、更改和删除表/索引	Create/Alter/Drop table Create/Alter/Drop index





续表

SQL 关键字	功　能
查询表中的信息	Select
获取数据库、表和查询的有关信息	Describe、Explain、Show
修改表中的信息	Delete、Insert、Update、Load data、Optimize table、Replace
管理语句	Flush、Grant、Kill、Revoke
其他语句	Create/Drop function、Lock/Unlock tables、Set

在 MySQL 中可以直接使用 SQL 语句，这些语句几乎可以不加修改地嫁入到 PHP 语言中。另外，MySQL 还允许在 SQL 语句中使用注释，有以下 3 种编写注释的方式：

☑ 以“#”号开头直到行尾的所有内容都是注释。

☑ 以“-- ”号开头直到行尾的所有内容都是注释。注意，在“--”后面还有一个空格。

☑ 以“/*”开始，以“*/”结束的所有内容都是注释，可以对多行进行注释。

7.2　MySQL 服务器的启动和关闭

视频讲解：配套资源\mr\7\video\MySQL 服务器的启动和关闭.exe

通过系统服务器和命令提示符（DOS）都可以启动和停止 MySQL，操作非常简单。但通常情况下，不要暂停或停止 MySQL 服务器，否则数据库将无法使用。

7.2.1　启动 MySQL 服务器

启动 MySQL 服务器的方法有两种：通过系统服务器和命令提示符（DOS）。

1. 通过系统服务器启动 MySQL

选择“开始”/“设置”/“控制面板”/“管理工具”/“服务”命令，打开“服务”窗口，从“名称”列中找到 MySQL 服务，单击鼠标右键，选择“启动”命令，如图 7.1 所示。



图 7.1　通过系统服务器启动 MySQL

2. 在命令提示符下启动 MySQL

选择“开始”/“运行”命令，输入 cmd，进入 DOS 窗口，在命令提示符下输入如下指令：

```
\> net start MySQL
```



按 Enter 键就会看到启动信息，如图 7.2 所示。



多学两招：

要想在命令提示符下操作 MySQL 服务器，前提是在本机的“计算机”/“系统属性”/“环境变量”/“系统变量”/“path”中已经完成 MySQL 启动文件夹（如 E:\xampp\mysql\bin）的加载操作，否则将不能通过命令操作 MySQL。添加环境变量的运行效果如图 7.3 所示。

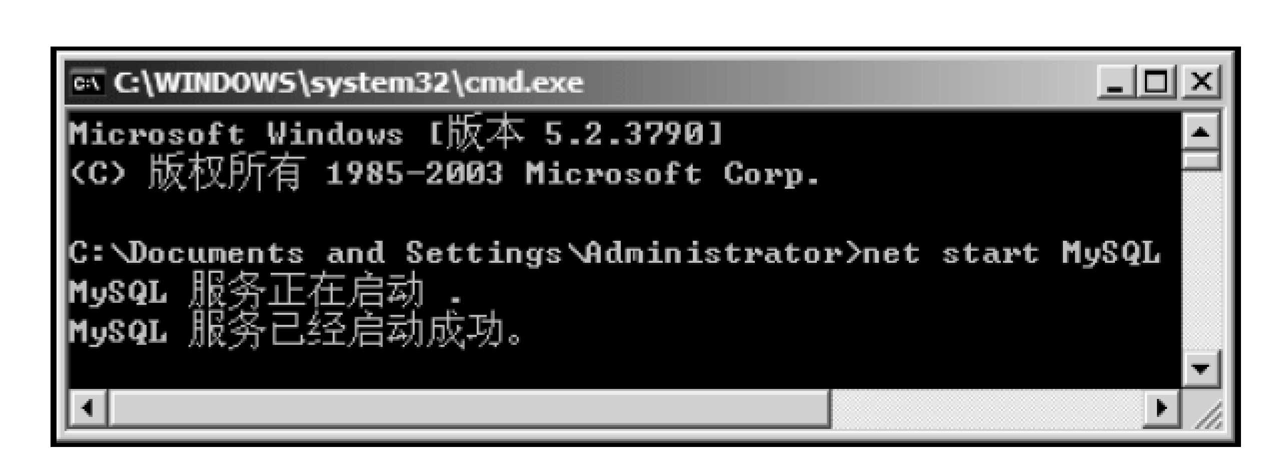


图 7.2　在命令提示符下启动 MySQL



图 7.3　设置系统变量

7.2.2　连接 MySQL 服务器

当 MySQL 服务器启动后就可以进行服务器的连接了。连接 MySQL 服务器也有两种方法：一种是进入 DOS 窗口，通过命令来连接；另一种是使用 MySQL 数据库系统函数连接。关于通过 MySQL 数据库系统函数连接数据库将在第 8 章中进行详细讲解，这里只介绍在命令提示符（DOS）中操作数据库。

选择“开始”/“运行”命令，输入 cmd，进入 DOS 窗口。在命令提示符下输入如下指令：

```
\> MySQL –uroot –h127.0.0.1 –ppassword
```

其中，-u 后输入的是用户名 root；-h 后输入的是 MySQL 数据库服务器地址；-p 后输入的是密码。

输入完命令语句后，按 Enter 键就进入到 MySQL 数据库中，如图 7.4 所示。

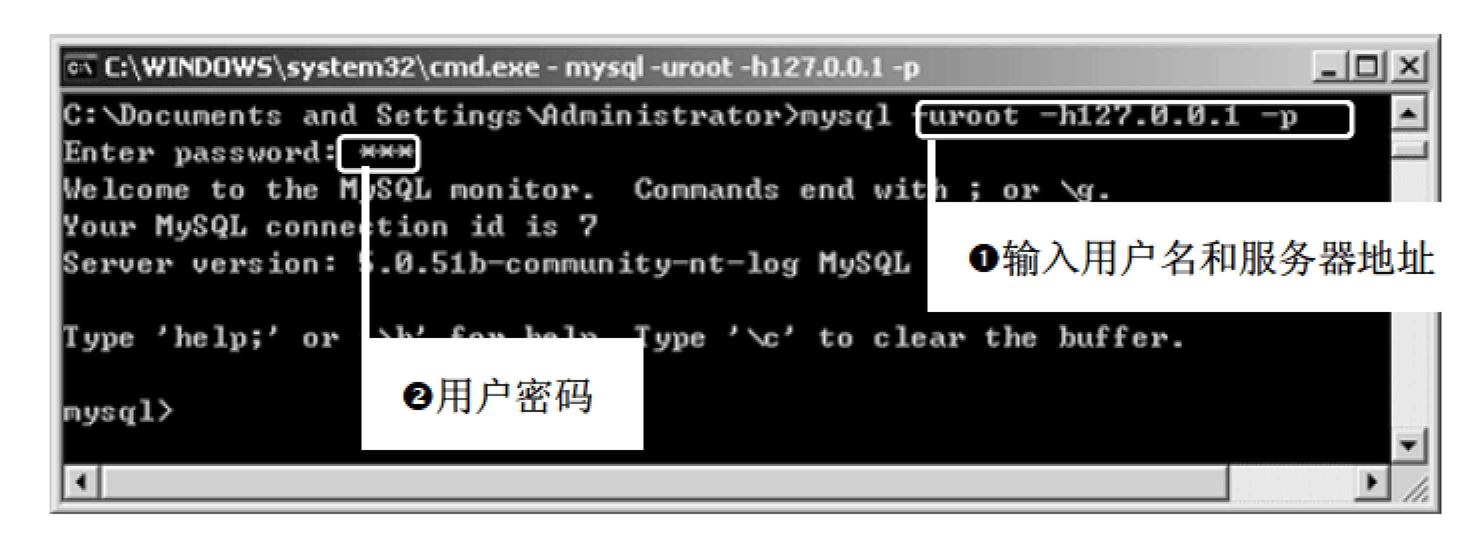
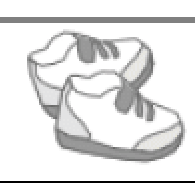
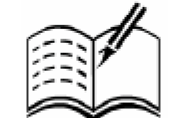
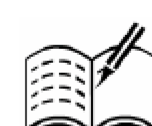


图 7.4　以隐藏密码方式连接服务器

脚下留神：

在输入的用户名“-uroot”与地址“-h127.0.0.1”之间必须有一个空格，同样在“-h127.0.0.1”与“-ppassword”之间也必须有一个空格，同时，在该命令的结束之处不必加“;”。





Note

多学两招：

为了保护 MySQL 数据库的密码，可以采用如图 7.4 所示的密码输入方式。如果密码在-p 后直接给出，那么密码就以明文显示，如 MySQL –u root –h 127.0.0.1 –p 111。按 Enter 键后再输入密码将以加密的方式显示，然后再按 Enter 键即可成功连接 MySQL 服务器。

7.2.3 关闭 MySQL 服务器

关闭 MySQL 服务器也可以通过系统服务器和命令提示符（DOS）两种方式操作。

1. 通过系统服务器关闭 MySQL

选择“开始”/“设置”/“控制面板”/“管理工具”/“服务”命令，打开“服务”窗口，从“名称”列中找到 MySQL 服务，单击鼠标右键，选择“停止”命令，如图 7.5 所示。

2. 在命令提示符下关闭 MySQL

选择“开始”/“运行”命令，输入 cmd，进入 DOS 窗口，在命令提示符下输入如下指令：

```
\> net stop MySQL
```

按 Enter 键后可看到服务停止信息，如图 7.6 所示。

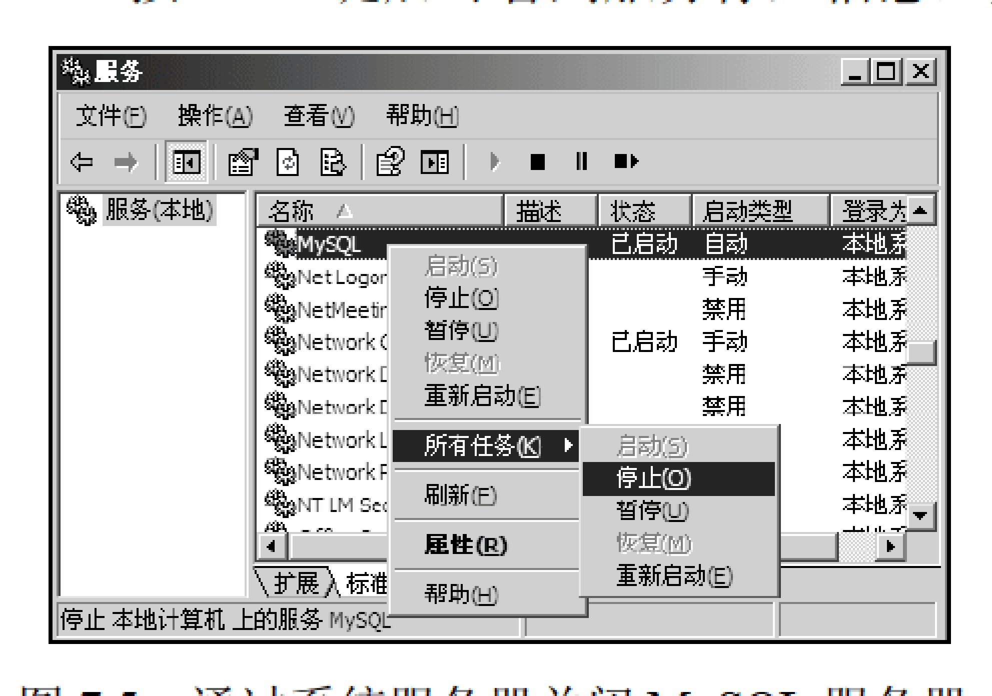


图 7.5 通过系统服务器关闭 MySQL 服务器



图 7.6 在命令提示符下关闭 MySQL 服务器

7.3 操作 MySQL 数据库

视频讲解：配套资源\mr\7\video\操作 MySQL 数据库.exe

针对 MySQL 数据库的操作可以分为创建、选择和删除 3 种。

7.3.1 创建新数据库

在 MySQL 中，应用 CREATE DATABASE 语句创建数据库。其语法格式如下：

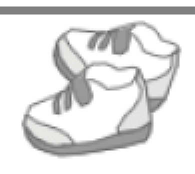
```
CREATE DATABASE db_name ;
```

其中，db_name 是要创建的数据库名称，该名称必须是合法的。对于数据库的命名，有如下规则：

☑ 不能与其他数据库重名。

☑ 名称可以由任意字母、阿拉伯数字、下划线（_）或“$”组成，可以使用上述的任意字符开头，但不能使用单独的数字，那样会造成名称与数值相混淆。

☑ 名称最长可由 64 个字符组成（还包括表、列和索引的命名），而别名最多可长达 256



个字符。

☑　不能使用 MySQL 关键字作为数据库/表名。

下面通过 CREATE DATABASE 语句创建一个名称为 db_database14 的数据库。运行结果如图 7.7 所示。

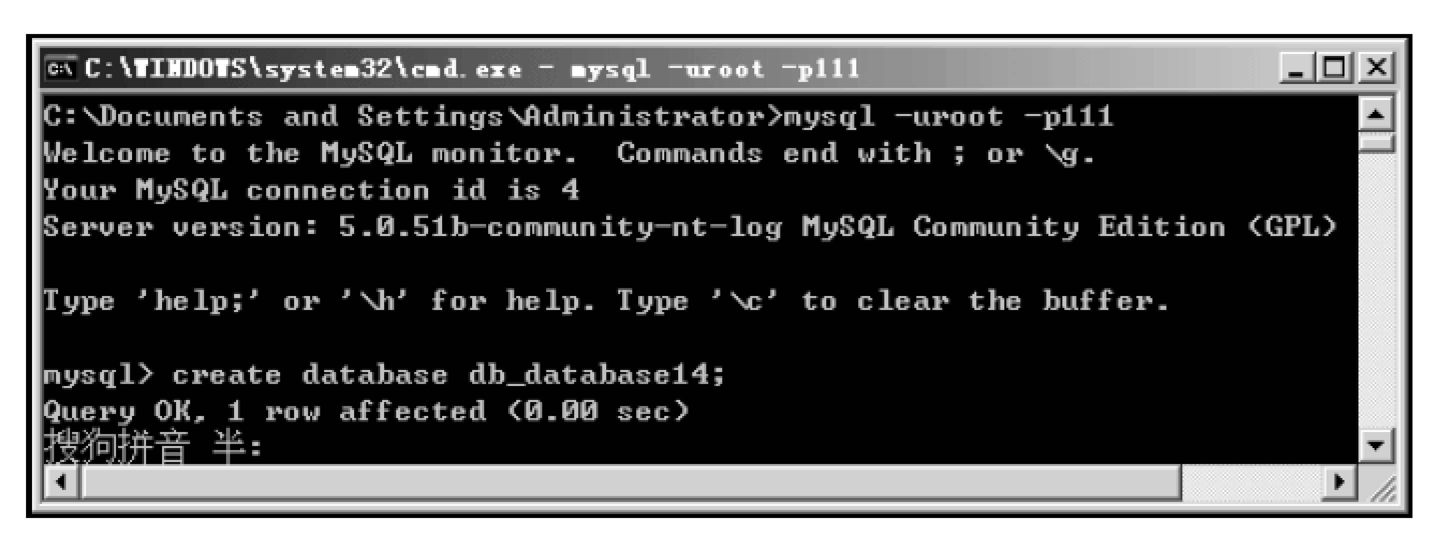


图 7.7　创建数据库

指点迷津：

在创建数据库时，首先连接 MySQL 服务器，用户名是 root，密码是 111，然后编写 “create database db_database14;” SQL 语句，数据库即创建成功。

7.3.2　选择指定数据库

USE 语句用于选择一个数据库，使其成为当前默认数据库。其语法如下：

```
USE db_name ;
```

例如，选择名称为 db_database14 的数据库，操作命令如图 7.8 所示。

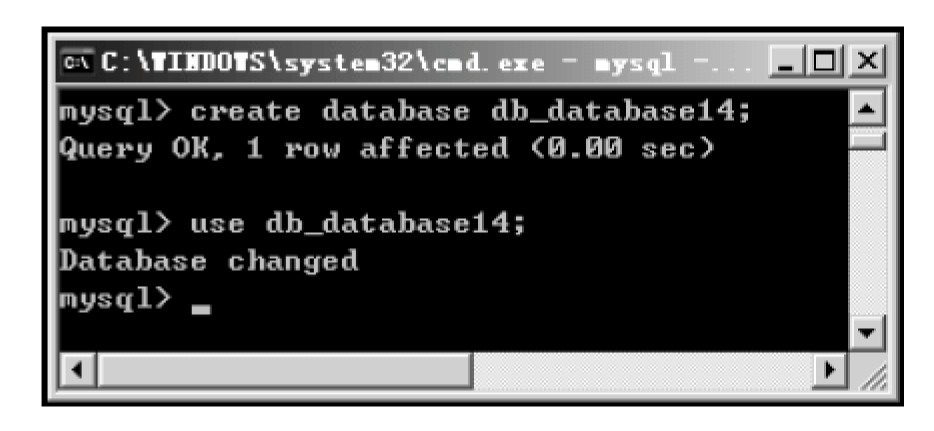


图 7.8　选择数据库

如图 7.8 所示，已经进入到 db_database14 数据库中。

7.3.3　删除指定数据库

删除数据库使用的是 DROP DATABASE 语句，其语法如下：

```
DROP DATABASE db_name ;
```

例如，通过 DROP DATABASE 语句删除名称为 db_database14 的数据库，操作命令如图 7.9 所示。

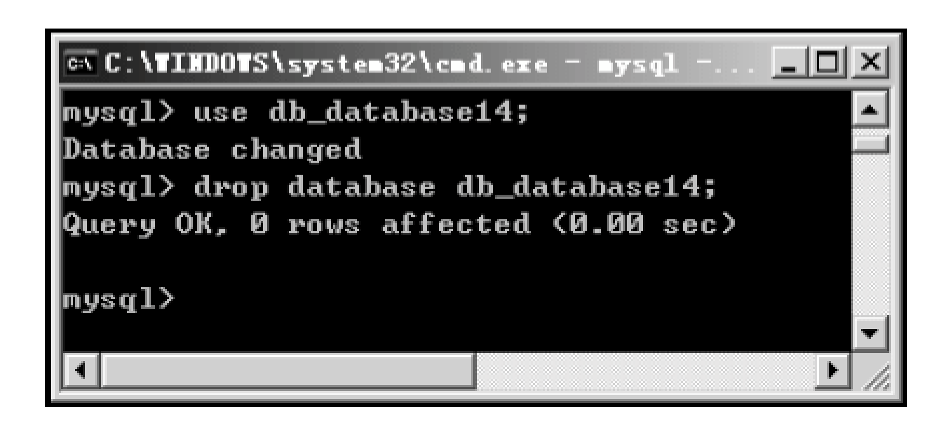
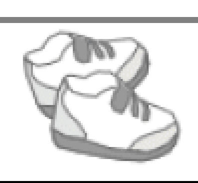


图 7.9　删除数据库



多学两招：

对于删除数据库的操作，用户应谨慎使用，一旦执行这项操作，数据库的所有结构和数据都会被删除，且没有恢复的可能，除非数据库有备份。



小测试

登录 MySQL 数据库后，可以通过 CREATE DATABASE 命令创建新的 MySQL 数据库。这里使用 CREATE DATABASE 命令创建名为 db_book 的图书数据库。其实现过程如下：

（1）进入命令提示符窗口，启动 MySQL 服务，连接 MySQL 服务器。

（2）通过数据库函数在命令提示符下编写命令。代码如下：

```
CREATE DATABASE db_book;
```

按 Enter 键，运行结果如图 7.10 所示。

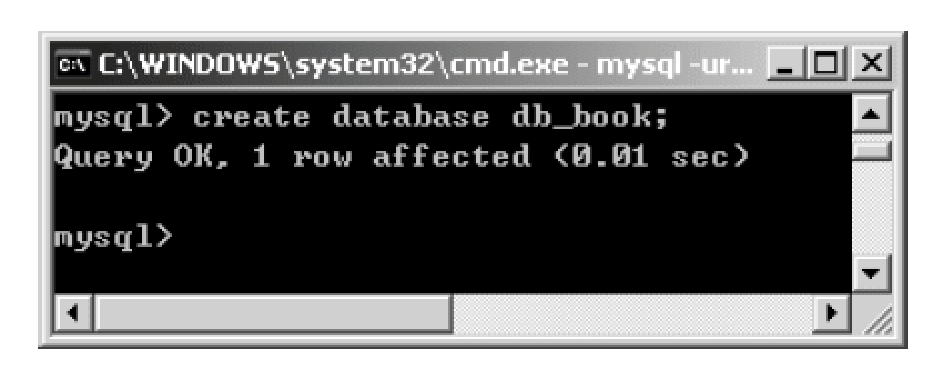


图 7.10　创建 MySQL 数据库

7.4　操作 MySQL 数据表

视频讲解：配套资源\mr\7\video\操作 MySQL 数据表.exe

MySQL 数据表的基本操作包括创建、查看、修改、重命名和删除。

7.4.1　创建一个表

创建数据表使用 CREATE TABLE 语句，其语法如下：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] 数据表名
[(create_definition,…)][table_options] [select_statement]
```

CREATE TABLE 语句的参数说明如表 7.3 所示。

表 7.3　CREATE TABLE 语句的参数说明

参　　数	说　　明
TEMPORARY	如果使用该关键字，表示创建一个临时表
IF NOT EXISTS	该关键字用于避免表存在时 MySQL 报告的错误
create_definition	这是表的列属性部分。MySQL 要求在创建表时，表要至少包含一列
table_options	表的一些特性参数
select_statement	SELECT 语句描述部分，使用它可以快速地创建表

下面介绍列属性 create_definition 部分，每一列定义的具体格式如下：

```
col_name   type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT]
           [PRIMARY KEY ] [reference_definition]
```

属性 create_definition 的参数说明如表 7.4 所示。







表 7.4 属性 create_definition 的参数说明

参 数	说 明
col_name	字段名
type	字段类型
NOT NULL | NULL	指出该列是否允许为空值，系统一般默认允许为空值，所以当不允许为空值时，必须使用 NOT NULL
DEFAULT default_value	表示默认值
AUTO_INCREMENT	表示是否为自动编号，每个表只能有一个 AUTO_INCREMENT 列，并且必须被索引
PRIMARY KEY	表示是否为主键。一个表只能有一个 PRIMARY KEY。如果表中没有 PRIMARY KEY，而某些应用程序需要 PRIMARY KEY，则 MySQL 将返回第一个没有任何 NULL 列的 UNIQUE 键，作为 PRIMARY KEY
reference_definition	为字段添加注释

例如，创建一个简单的数据表，使用 CREATE TABLE 语句在 MySQL 数据库 db_db_database14 中创建一个名为 tb_admin 的数据表，该表包括 id、user、password 和 createtime 4 个字段。创建的 SQL 语句如下：

```
CREATE TABLE `tb_admin` (
`id` INT( 4 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`user` VARCHAR( 50 ) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL ,
`password` VARCHAR( 50 ) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL ,
`createtime` DATETIME NOT NULL ) ENGINE = MYISAM CHARACTER SET utf8 COLLATE
utf8_unicode_ci;
```

在命令模式下的运行结果如图 7.11 所示。

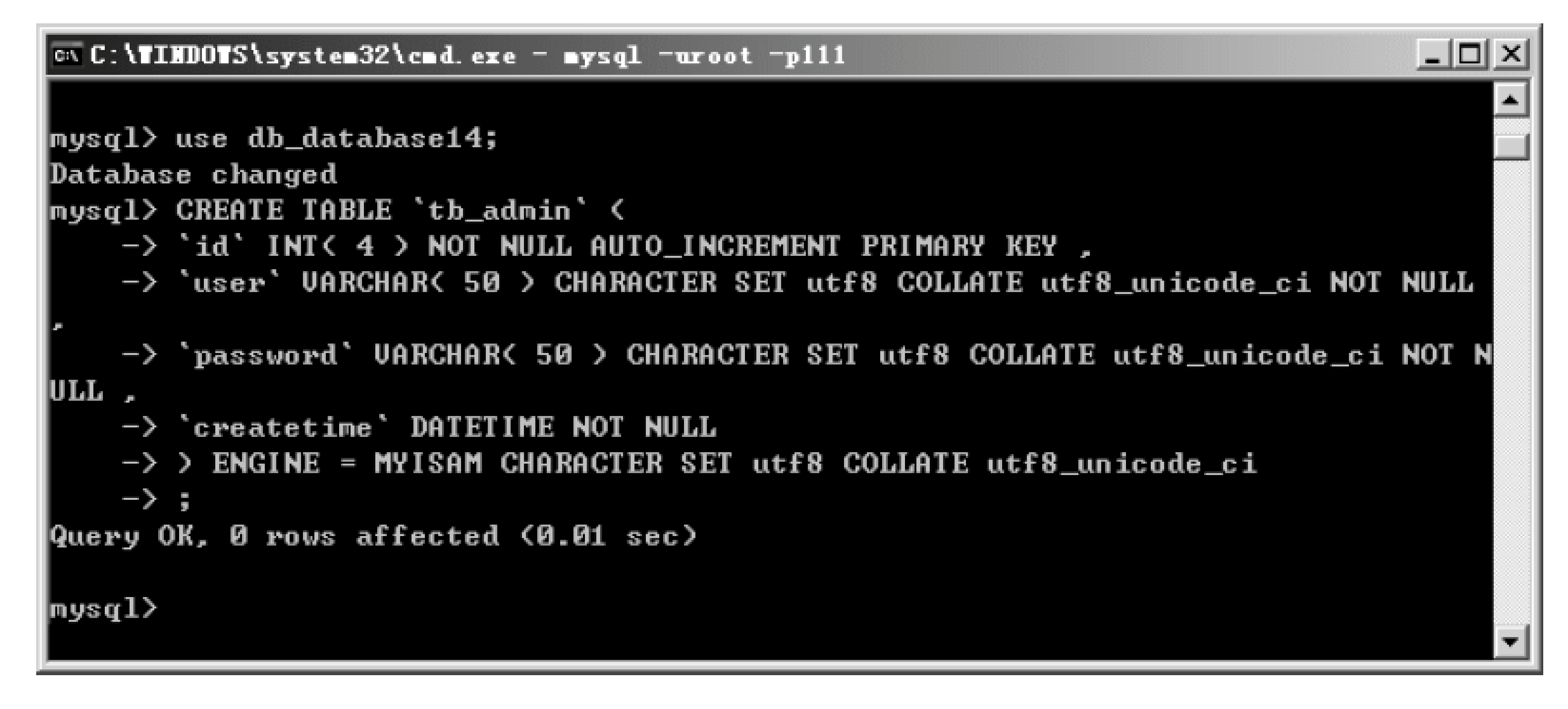


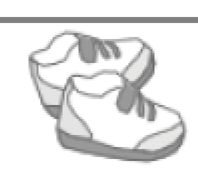
图 7.11 创建 MySQL 数据库

多学两招：

在输入 SQL 语句时，可以一行全部输出，也可以每个字段都换行输出，这里建议用换行输出，这样看上去更加美观、易懂，在语句出现错误时也更容易查找。

7.4.2 查看数据表结构

对于一个创建成功的数据表，可以使用 SHOW COLUMNS 语句或 DESCRIBE 语句查看数



Note

据表的结构。

1. SHOW COLUMNS 语句

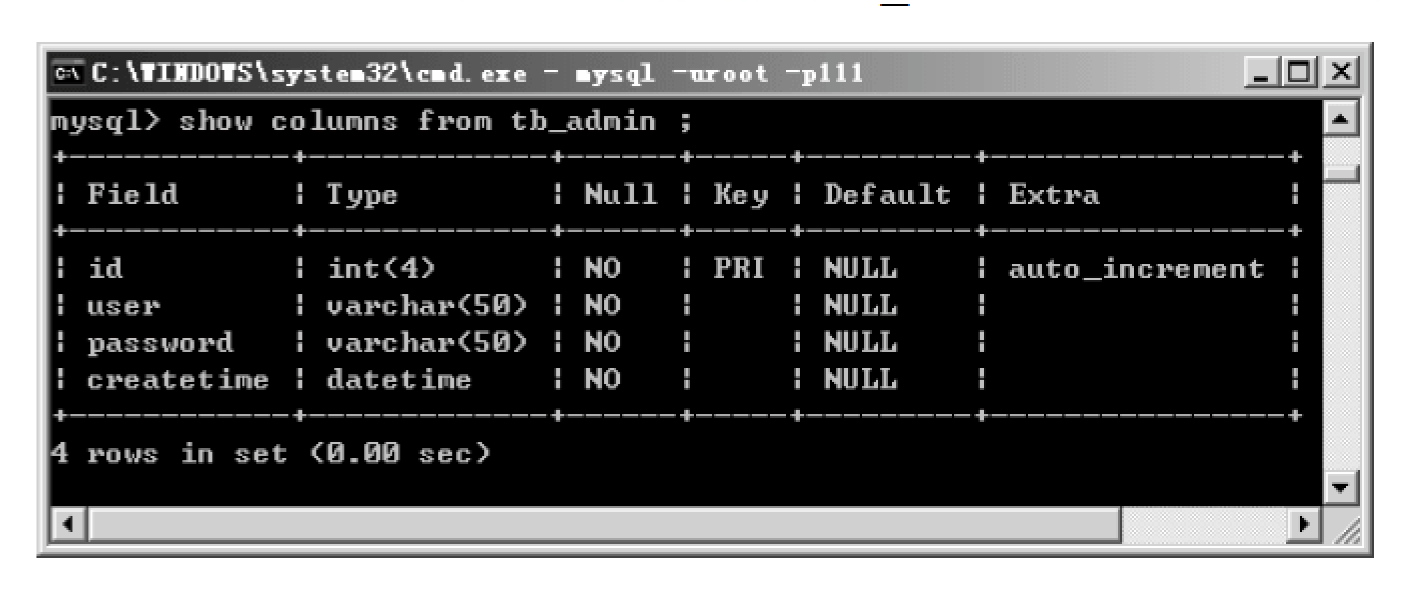
SHOW COLUMNS 语句查看一个指定的数据表，其语法如下：

```
SHOW  [FULL] COLUMNS  FROM 数据表名 [FROM 数据库名];
```

或写成：

```
SHOW  [FULL] COLUMNS  FROM 数据表名.数据库名;
```

例如，使用 SHOW COLUMNS 语句查看数据表 tb_admin 的结构，如图 7.12 所示。



```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p111
mysql> show columns from tb_admin ;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(4)      | NO   | PRI | NULL    | auto_increment |
| user       | varchar(50) | NO   |     | NULL    |                |
| password   | varchar(50) | NO   |     | NULL    |                |
| createtime | datetime    | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

图 7.12　查看表结构

2. DESCRIBE 语句

DESCRIBE 语句的语法如下：

```
DESCRIBE 数据表名;
```

其中，DESCRIBE 可以简写成 DESC。在查看表结构时，也可以只列出某一列的信息。其语法如下：

```
DESCRIBE 数据表名 列名;
```

例如，使用 DESCRIBE 语句的简写形式查看数据表 tb_admin 中的某一列信息，如图 7.13 所示。



```
C:\WINDOWS\system32\cmd.exe - m
mysql> desc tb_admin user;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| user  | varchar(30) | NO   |     |         |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> _
```


图 7.13　查看表的某一列信息

7.4.3　修改数据表结构

修改数据表结构使用 ALTER TABLE 语句，具体操作包括：增加或删除字段、修改字段名称或字段类型、设置取消主键外键、设置取消索引以及修改表的注释等。其语法如下：

```
Alter[IGNORE] TABLE 数据表名 alter_spec[,alter_spec]…
```

其中，alter_spec 子句定义要修改的内容，其语法如下：

```
alter_specification:
    ADD [COLUMN] create_definition [FIRST | AFTER column_name ]   //添加新字段
```






```
    | ADD INDEX [index_name] (index_col_name,...)                    //添加索引名称
    | ADD PRIMARY KEY (index_col_name,...)                           //添加主键名称
    | ADD UNIQUE [index_name] (index_col_name,...)                   //添加唯一索引
    | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}//修改字段名称
    | CHANGE [COLUMN] old_col_name create_definition                 //修改字段类型
    | MODIFY [COLUMN] create_definition                              //修改子句定义字段
    | DROP [COLUMN] col_name                                         //删除字段名称
    | DROP PRIMARY KEY                                               //删除主键名称
    | DROP INDEX index_name                                          //删除索引名称
    | RENAME [AS] new_tbl_name                                       //更改表名
    | table_options
```

ALTER TABLE 语句允许指定多个动作，其动作间使用逗号分隔，每个动作表示对表的一个修改。



多学两招：

在使用 ALTER TABLE 语句修改数据表时，如果指定 IGNORE 参数，当出现重复的行时，则只执行一行，其他重复的行将被删除。

例如，添加一个新的字段 email，类型为 varchar(50)，not null，将字段 user 的类型由 varchar(50)改为 varchar(80)，代码如下：

```
alter table tb_admin add email varchar(50) not null ,modify user varchar(80);
```

在命令模式下的运行情况如图 7.14 所示。

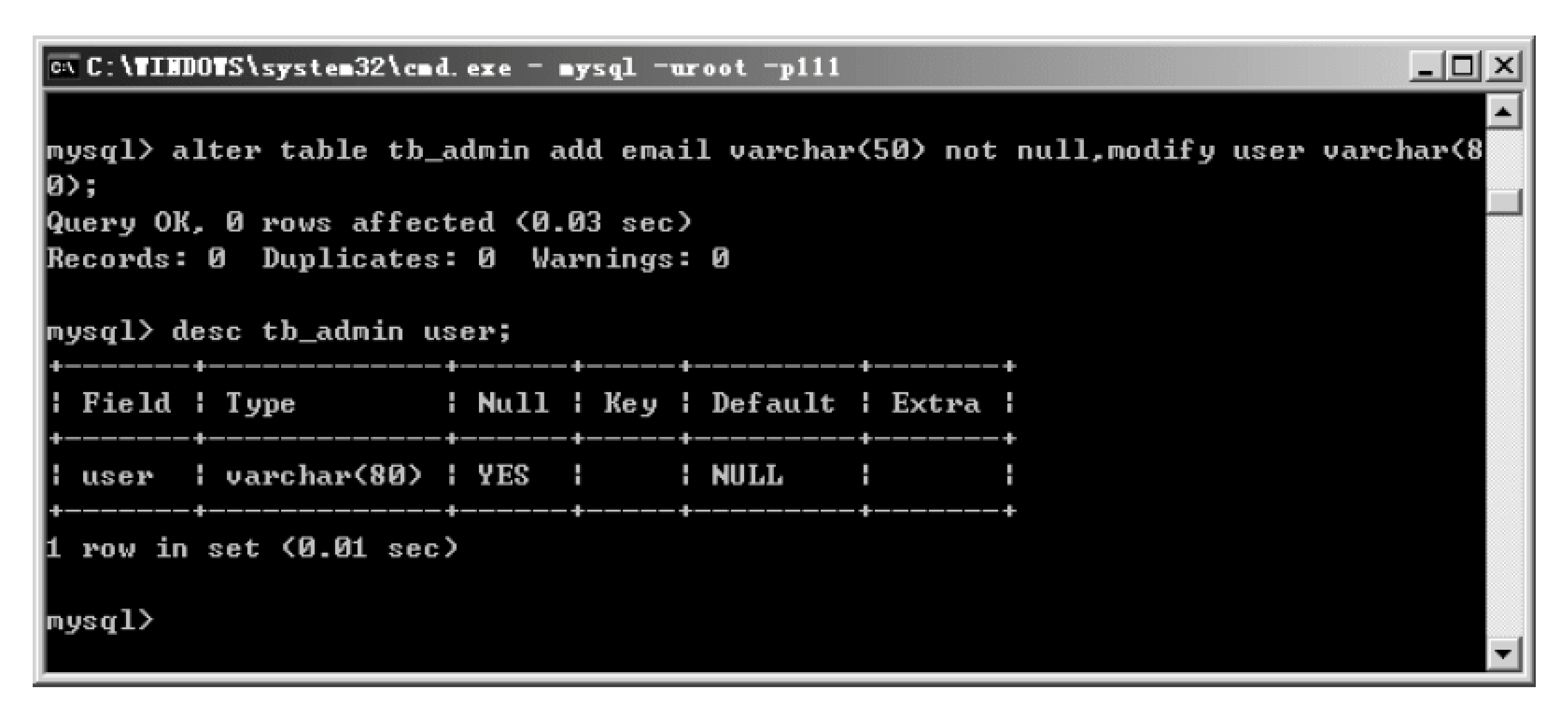


图 7.14　修改表结构

图 7.14 中只给出了修改 user 字段类型的结果，读者可以通过语句 MySQL> show tb_admin;查看整个表的结构，以确认 email 字段是否添加成功。

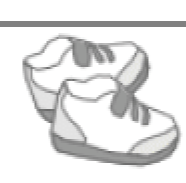
脚下留神：

通过 alter 修改表列的前提是必须将表中的数据全部删除。

7.4.4　重命名数据表

重命名数据表使用 RENAME TABLE 语句，其语法如下：

```
RENAME TABLE  数据表名1 To  数据表名2
```



多学两招：

RENAME TABLE 语句可以同时对多个数据表进行重命名，多个表之间以逗号“,”分隔。



例如，对数据表 tb_admin 进行重命名，更名后的数据表为 tb_user，如图 7.15 所示。



图 7.15　对数据表进行重命名

7.4.5　删除指定数据表

删除指定数据表使用 DROP TABLE 语句，其语法如下：

```
DROP TABLE 数据表名;
```

例如，删除数据表 tb_user，如图 7.16 所示。

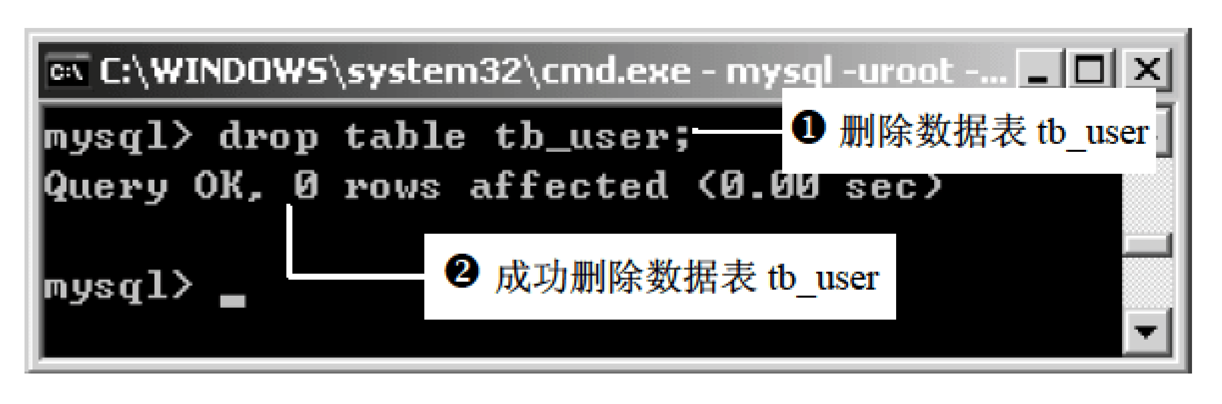


图 7.16　删除指定数据表

在删除数据表的过程中删除一个不存在的表将会产生错误，如果在删除语句中加入 IF EXISTS 关键字则不会出错。其语法如下：

```
drop table if exists 数据表名;
```

脚下留神：

对于删除数据表的操作，读者应谨慎使用。一旦删除数据表，那么表中的数据将会全部清除，若没有备份则无法恢复。

多学两招：

在执行 CREATE TABLE、ALTER TABLE 和 DROP TABLE 中的任何操作时，首先必须要选择数据库，否则是无法对数据表进行操作的。

☎小测试

通过 CREATE TABLE 语句在图书数据库中创建图书信息表，操作命令如图 7.17 所示。



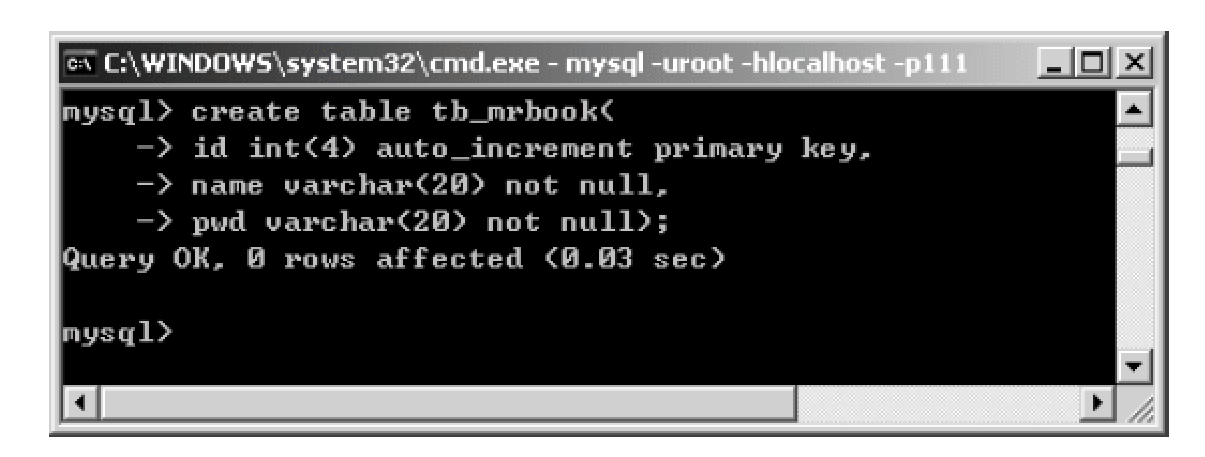


图 7.17　数据表创建成功



Note

（1）启动 MySQL 服务，建立 MySQL 服务器连接。

（2）使用 use db_book;语句设定数据库标记，这里设定的数据库是 db_book。

（3）通过 create table tb_mrbook();语句创建数据表，代码如下：

```
MySQL> use db_book;
Database changed
MySQL> create table tb_mrbook(
    -> id int(4) auto_increment primary key,
    -> name varchar(20) NOT NULL,
    -> pwd varchar(20) NOT NULL);
Query OK, 0 rows affected (0.05 sec)
```

（4）断开 MySQL 服务器连接，结束 MySQL 服务。

脚下留神：

在创建数据表的过程中，一定要注意尖括号（<）和括号的区别。若将二者混淆，将会导致数据表创建失败。

7.5　操作 MySQL 数据

视频讲解：配套资源\mr\7\video\操作 MySQL 数据.exe

数据库中包含数据表，而数据表中包含数据。在 MySQL 与 PHP 的结合应用中，真正被操作的是数据表中的数据，因此如何更好地操作和使用这些数据才是使用 MySQL 数据库的根本。

7.5.1　向数据表中添加数据（INSERT）

建立一个数据库并在数据库中创建一个数据表，首先要向数据表中添加数据。这项操作可以通过 INSERT 语句来完成。

下面通过向 db_database14 数据库的 tb_mrbook 表中添加一条记录，介绍 INSERT 语句的 3 种语法。

（1）列出新添加数据的所有的值。其语法如下：

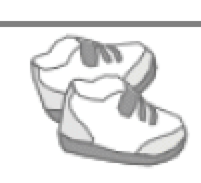
```
insert into table_name values (value1,value2, … )
```

（2）给出要赋值的列，然后再给出值。其语法如下：

```
insert into table_name (column_name,column_name2, … ) values (value1, value1, … )
```

（3）用 col_name = value 的形式给出列和值。其语法如下：

```
insert into table_name set column_name1 = value1,column_name2 = value2, …
```





Note

指点迷津：

采用第一种语法格式的优点是输入的 SQL 语句短小，且错误查找方便；缺点是如果字段很多，则不容易辨别数据属于哪一列，不易匹配。

多学两招：

数据的批量添加：LOAD DATA 和 MySQLIMPORT 语句。

LOAD DATA 语句，通过读取本地文件系统上的文件，可以将大量数据添加到数据库中。语法如下：

```
load data local infile  "filename.txt"  into table table_name
```

其中，filename.txt 是当前目录中的数据文件的名称；table_name 是要被装载的数据表。

LOAD DATA 语句读取客户机当前目录中数据文件 filename.txt 的内容，并将其内容装载到数据表 table_name 中。

MySQLIMPORT 语句直接从文件读取批量数据，它相当于 LOAD DATA 语句的一个接口。语法如下：

```
%MySQLimport    local table_name filename.txt ;
```

MySQLIMPORT 可以自动生成一个 LOAD DATA 语句，该语句把 filename.txt 文件中的数据装入 table_name 表中。MySQLIMPORT 根据文件名导出表名，即将文件名第一个圆点（.）前的所有字符作为表名。例如，文件 class.txt 被装入 class 表中。

7.5.2　更新数据表中的数据（UPDATE）

更新数据表中的数据使用 UPDATE 语句，语法如下：

```
update table_name
set column_name = new_value1,column_name2 = new_value2, …
where condition
```

其中，table_name 是更新的表名称；SET 子句指出要修改的列和它们给定的值；WHERE 子句是可选的，如果应用它，将指定记录中的哪一行应该被更新；否则，所有的记录行都将被更新。

例如，下面将管理员信息表 tb_user 中用户名为 tsoft 的管理员密码 111 修改为 123456，如图 7.18 所示。

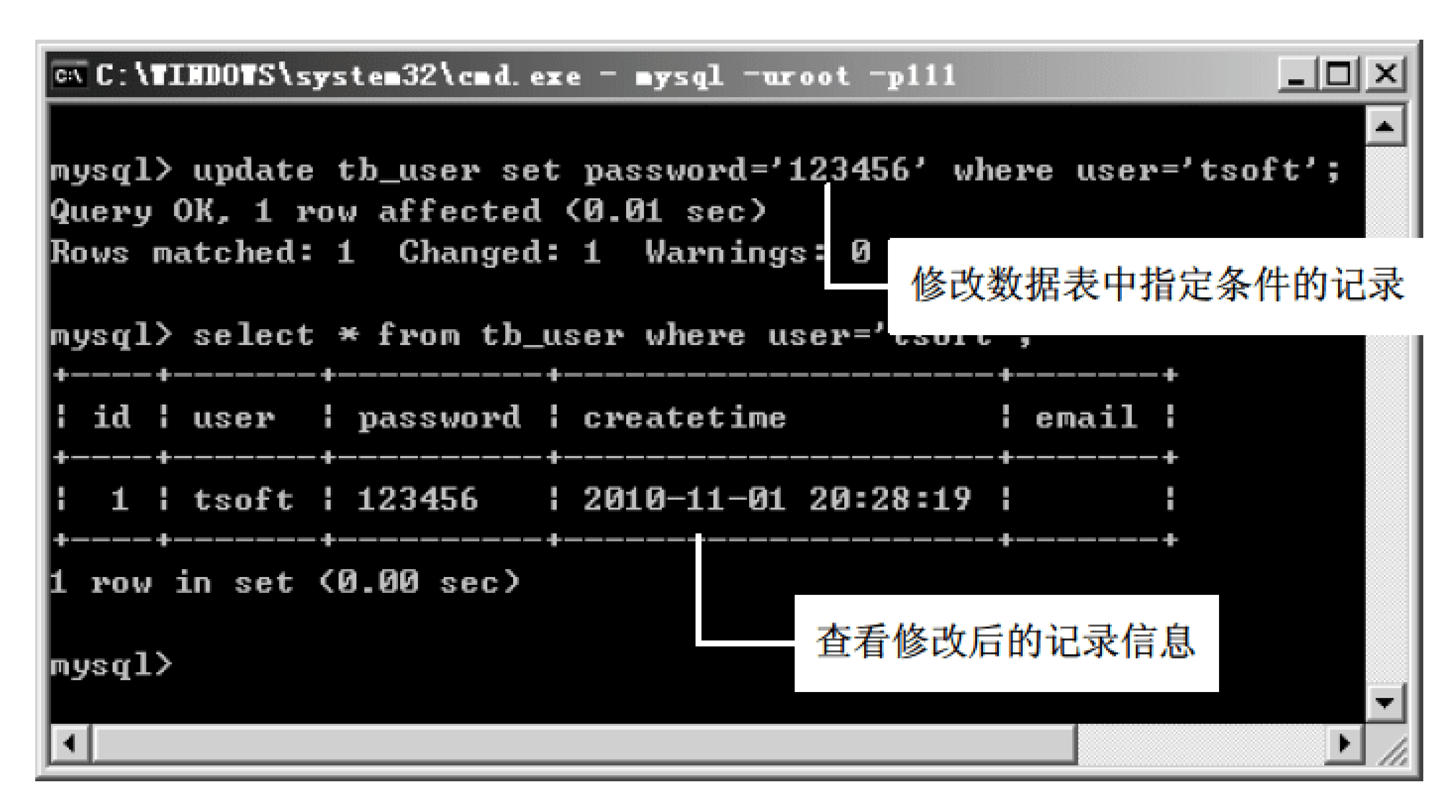
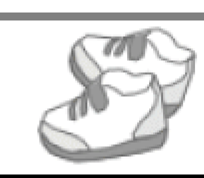


图 7.18　修改指定条件的记录





脚下留神：

在对数据表中的数据进行更新时一定要保证 WHERE 子句的正确性，一旦 WHERE 子句出错，将会破坏所有改变的数据。

7.5.3　删除数据表中的数据（DELETE）

删除数据表中的数据使用 DELETE 语句，语法如下：

```
delete from table_name
where condition
```

该语句在执行过程中删除 table_name 表中的记录，如果没有指定 WHERE 条件，将删除所有的记录；如果指定 WHERE 条件，将按照指定的条件进行删除。

例如，删除管理员数据表 tb_user 中用户名为 mr 的记录信息，如图 7.19 所示。



图 7.19　删除数据表中指定的记录

多学两招：

（1）在实际应用中，当执行删除操作时，执行删除的条件一般应为数据的 id，而不是具体某个字段值，这样可以避免出现一些不必要的错误。（2）使用 DELETE 语句删除整个表的效率并不高，建议读者使用 TRUNCATE 语句，它可以快速地删除表中所有的内容。

7.5.4　查询数据表中的数据（SELECT）

创建数据库的目的不仅是保存数据，更重要的是为了使用其中的数据。使用 SELECT 查询语句可以从数据库中把数据查询出来。语法如下：

```
select selection_list                //要查询的内容，选择哪些列
from table_list                      //从什么表中查询，从何处选择行
where primary_constraint             //查询时需要满足的条件，行必须满足的条件
group by grouping_columns            //如何对结果进行分组
order by sorting_cloumns             //如何对结果进行排序
having secondary_constraint          //查询时满足的第二条件
limit count                          //限定输出的查询结果
```

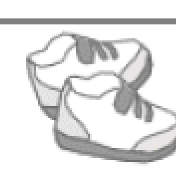
下面对 SELECT 查询语句的参数进行详细讲解。

1. selection_list

设置查询内容。如果要查询表中所有列，则可以将其设置为“*”；如果要查询表中某一列或多列，则直接输入列名，并以“,”作为分隔符。

例如，查询 tb_mrbook 数据表中所有列和查询 user 和 pass 列的代码如下：

```
select * from tb_mrbook;             //查询数据表中所有数据
select user,pass from tb_mrbook;     //查询数据表中user和pass列的数据
```





Note

2. table_list

指定查询的数据表。既可以从一个数据表中查询，也可以从多个数据表中进行查询，多个数据表之间用“,”进行分隔，并且通过 where 子句使用连接运算来确定表之间的联系。

例如，从 tb_mrbook 和 tb_bookinfo 数据表中查询“bookname='PHP 编程宝典'”的作者和价格，其代码如下：

```
select tb_mrbook.id,tb_mrbook.bookname,
    -> author,price from tb_mrbook,tb_bookinfo
    -> where tb_mrbook.bookname = tb_bookinfo.bookname and
    -> tb_bookinfo.bookname = 'php入门与精通';
```

在上面的 SQL 语句中，因为两个表都有 id 字段和 bookname 字段，为了告诉服务器要显示的是哪个表中的字段信息，应加上前缀。语法如下：

```
表名.字段名
```

tb_mrbook.bookname = tb_bookinfo.bookname 将表 tb_mrbook 和 tb_bookinfo 连接起来，称为等同连接；如果不使用 tb_mrbook.bookname = tb_bookinfo.bookname，那么产生的结果将是两个表的笛卡儿积，称为全连接。

3. WHERE 条件语句

在使用查询语句时，如果要从很多的记录中查询出想要的记录，就需要一个查询的条件。只有设定了查询的条件，查询才有实际的意义。设定查询条件应使用的是 WHERE 子句。

WHERE 子句的功能非常强大，通过它可以实现很多复杂的条件查询。在使用 WHERE 子句时，需要使用一些比较运算符，常用的比较运算符如表 7.5 所示。

表 7.5　WHERE 子句常用的比较运算符

运　算　符	名　　称	示　　例	运　算　符	名　　称	示　　例
=	等于	id=5	is not null	n/a	id is not null
>	大于	id>5	between	n/a	id between1 and 15
<	小于	id<5	in	n/a	id in (3,4,5)
=>	大于等于	id=>5	not in	n/a	name not in (shi,li)
<=	小于等于	id<=5	like	模式匹配	name like ('shi%')
!=或<>	不等于	id!=5	not like	模式匹配	name not like ('shi%')
Is null	n/a	id is null	regexp	常规表达式	name 正则表达式

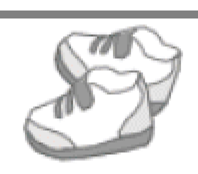
表 7.5 中列举的是 WHERE 子句常用的比较运算符，示例中的 id 是记录的编号，name 是表中的用户名。

例如，应用 WHERE 子句查询 tb_mrbook 表，条件是 type（类别）为 PHP 的所有图书，代码如下：

```
select * from tb_mrbook where type = 'php';
```

4. GROUP BY

通过 GROUP BY 子句可以将数据划分到不同的组中，实现对记录的分组查询。在查询时，所查询的列必须包含在分组的列中，目的是使查询到的数据没有矛盾。与 avg()或 sum()函数同时使用，GROUP BY 子句能发挥最大作用。



例如，查询表 tb_mrbook，按照 type 进行分组，求每类图书的平均价格，代码如下：

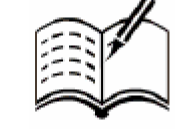
```
select bookname,avg(price),type from tb_mrbook group by type;
```

5. DISTINCT

使用 DISTINCT 关键字可以去除结果中重复的行。

例如，查询表 tb_mrbook，并在结果中去掉类型字段 type 中的重复数据，代码如下：

```
select distinct type from tb_mrbook;
```



6. ORDER BY

使用 ORDER BY 可以对查询的结果进行升序和降序（DESC）排列，在默认情况下，ORDER BY 按升序输出结果。如果要按降序排列，则可以使用 DESC 来实现。

当对含有 NULL 值的列进行排序时，如果是按升序排列，则 NULL 值将出现在最前面；如果是按降序排列，则 NULL 值将出现在最后。

例如，查询表 tb_mrbook 中的所有信息，按照 id 进行降序排列，并且只显示 3 条记录。其代码如下：

```
select * from tb_mrbook order by id desc limit 3;
```

7. LIKE

LIKE 属于较常用的比较运算符，通过它可以实现模糊查询。它有两种通配符："%"和下划线"_"。"%"可以匹配一个或多个字符，而"_"只匹配一个字符。

例如，查找所有第二个字母是"h"的图书，代码如下：

```
select * from tb_mrbook where bookname like('_h%');
```

8. CONCAT()函数

使用 CONCAT()函数可以联合多个字段，构成一个总的字符串。

例如，把 tb_mrbook 表中的书名（bookname）和价格（price）合并到一起，构成一个新的字符串。代码如下：

```
select id,concat(bookname,":",price) as info,f_time,type from tb_mrbook;
```

其中，合并后的字段名为 CONCAT()函数形成的表达式"concat(bookname,":",price)"，看上去十分复杂，通过 AS 关键字为合并字段取一个别名，使其看上去较为清晰一些。

9. LIMIT

LIMIT 子句可以对查询结果的记录条数进行限定，控制它输出的行数。

例如，查询表 tb_mrbook，按照图书价格降序排列，显示 3 条记录，代码如下：

```
select * from tb_mrbook order by price desc limit 3;
```

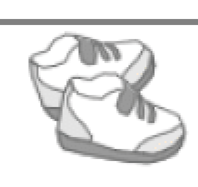
使用 LIMIT 还可以从查询结果的中间部分取值。首先要定义两个参数，参数 1 是开始读取的第一条记录的编号（在查询结果中，第一个结果的记录编号是 0，而不是 1）；参数 2 是要查询记录的个数。

例如，查询表 tb_mrbook，从编号 1 开始（即从第 2 条记录），查询 4 个记录，代码如下：

```
select * from tb_mrbook where id limit 1,4;
```

10. 使用函数和表达式

在 MySQL 中还可以使用表达式来计算各列的值，作为输出结果。另外，表达式还可以包



含一些函数。

例如，计算 tb_mrbook 表中各类图书的总价格，代码如下：

```
select sum(price) as total,type from tb_mrbook group by type;
```





在对 MySQL 数据库进行操作时，有时需要对数据库中的记录进行统计，例如求平均值、最小值、最大值等，这时可以使用 MySQL 中的统计函数。常用统计函数如表 7.6 所示。



表 7.6　常用统计函数

名　称	说　明
avg（字段名）	获取指定列的平均值
count（字段名）	如果指定了一个字段，则会统计出该字段中的非空记录。如果在前面增加 DISTINCT，则会统计不同值的记录，相同的值将被当作一条记录。如果使用 COUNT（*），则会统计包含空值的所有记录数
min（字段名）	获取指定字段的最小值
max（字段名）	获取指定字段的最大值
std（字段名）	指定字段的标准背离值
stdtev（字段名）	与 STD 相同
sum（字段名）	指定字段所有记录的总和

除了使用函数之外，还可以使用算术运算符、字符串运算符以及逻辑运算符来构成表达式。

例如，可以计算图书打 8 折之后的价格，代码如下：

```
select *, (price * 0.8) as '80%' from tb_mrbook;
```

☎小测试

1. 向表 tb_mrbook 中添加数据

通过 SQL 语句向图书信息表中添加数据，运行结果如图 7.20 所示。

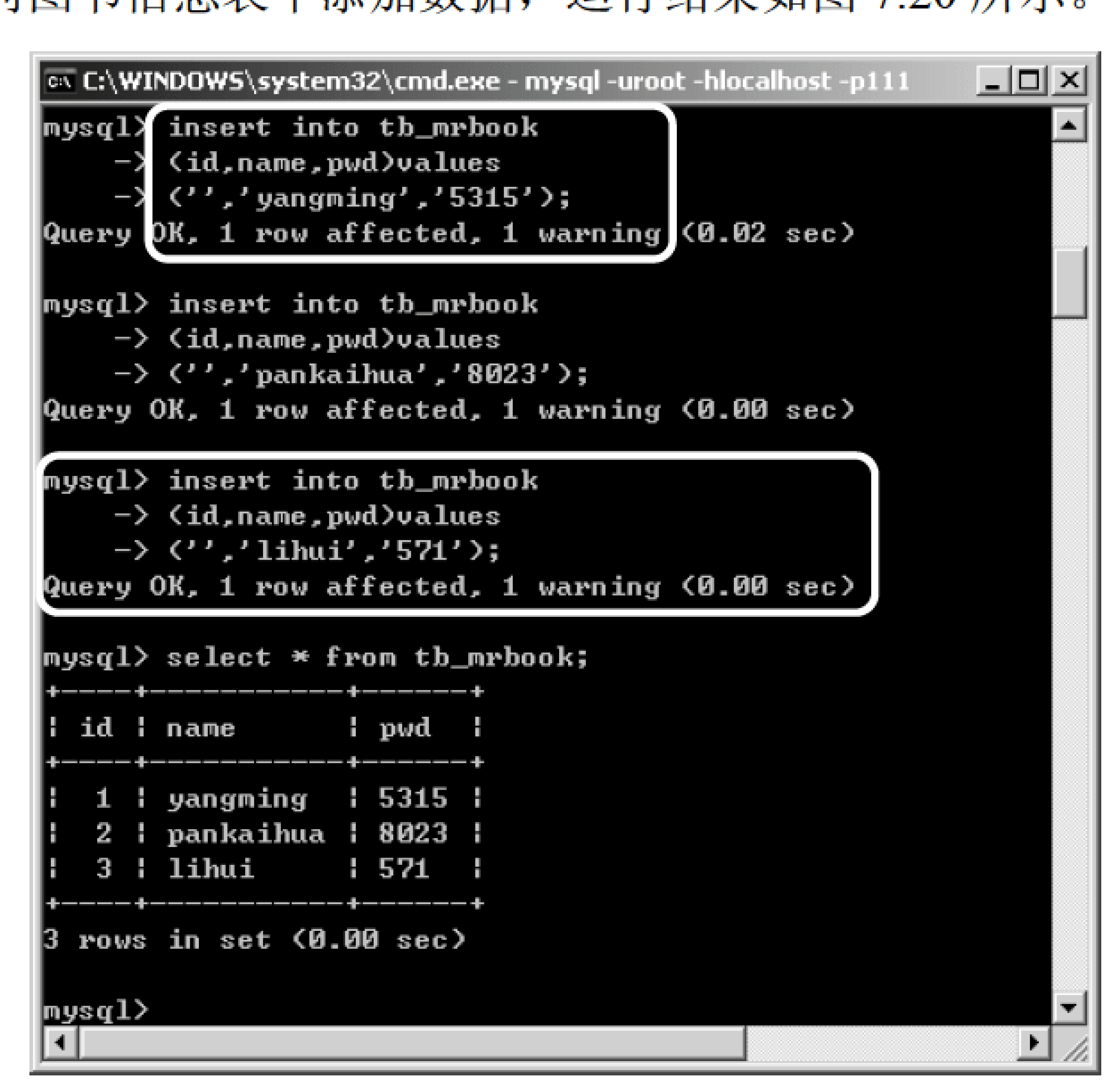


图 7.20　向图书信息表中添加数据





实现过程如下：

（1）在命令提示符窗口下开启 MySQL 服务，通过命令行语句连接 MySQL 服务器，选择数据库和数据表。

（2）编辑 INSERT 语句，执行数据的添加操作，其代码如下：

```
insert into tb_mrbook_php(id,name,pwd,last_date)values ('','lizhonghua','5315',now());
```

（3）当提示符窗口出现如图 7.21 所示的内容时，表示数据添加成功。

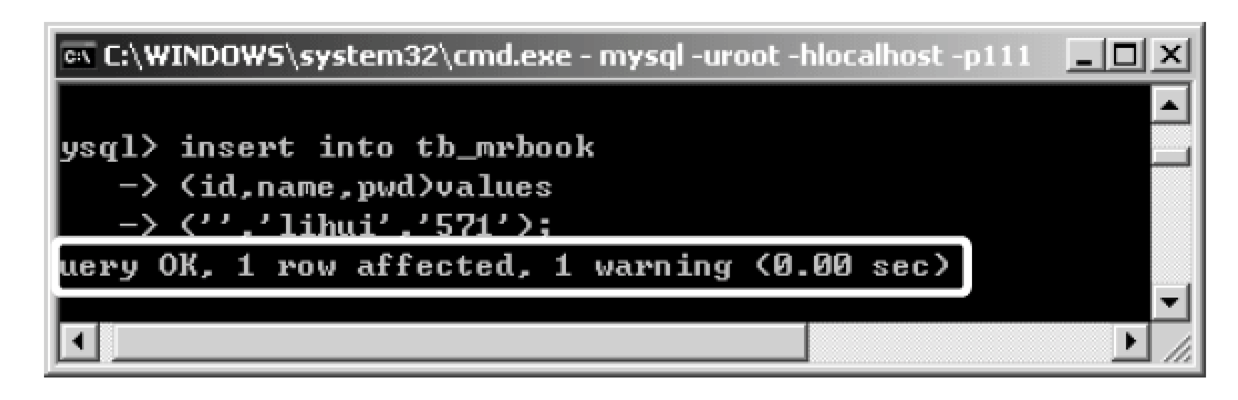


图 7.21　数据添加成功

（4）通过 SELECT 语句查询数据表中内容，代码如下（显示结果如图 7.21 所示）：

```
select * from tb_mrbook;
```

（5）断开 MySQL 服务器连接，关闭 MySQL 服务。

2. 修改表 tb_mrbook 中的用户的密码

通过修改语句对图书信息表中的用户密码进行修改，运行结果如图 7.22 所示。

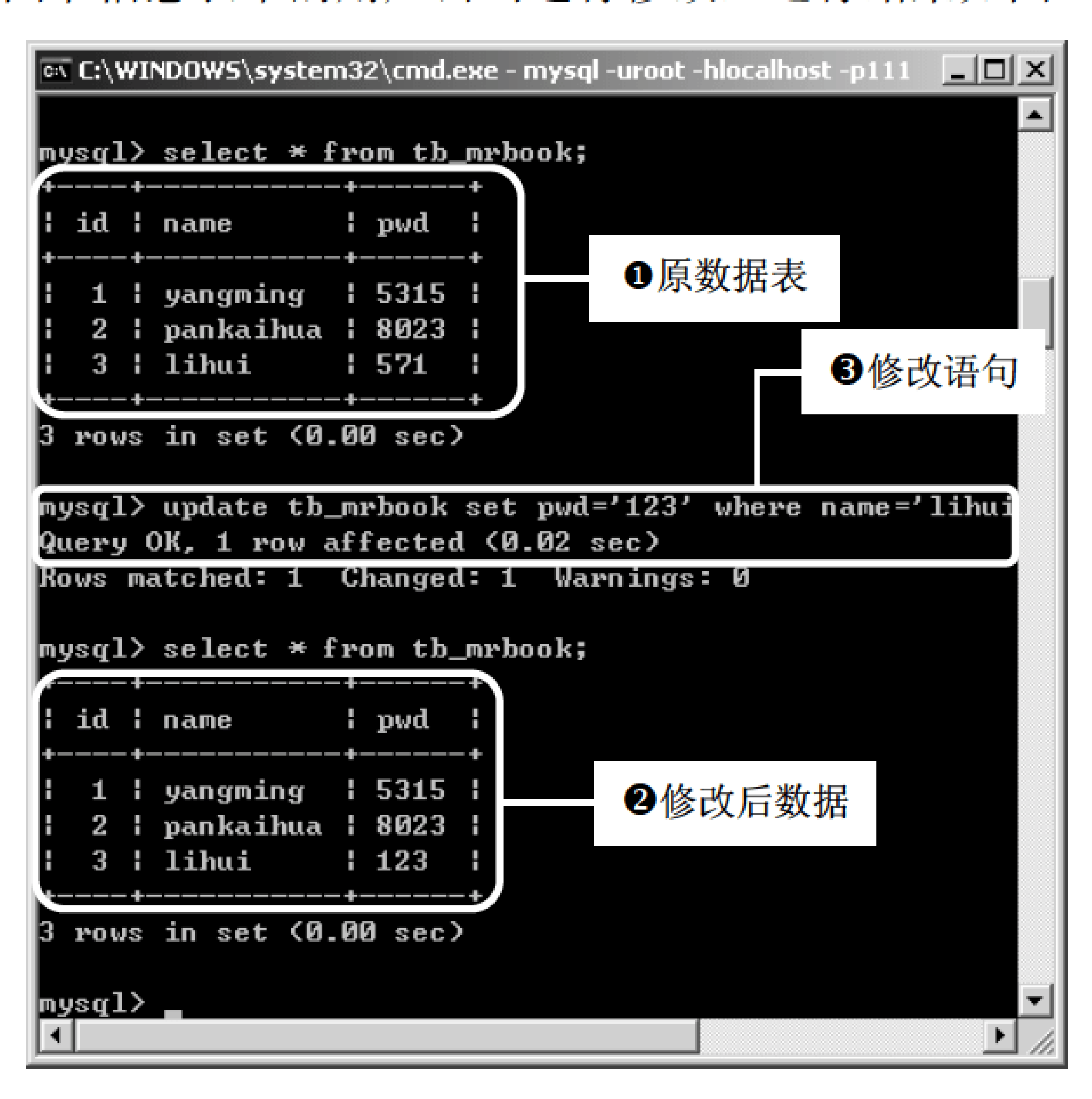


图 7.22　修改图书信息表

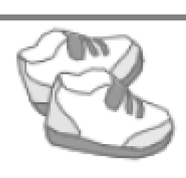
实现过程如下：

（1）在命令提示符窗口下开启 MySQL 服务，通过命令行语句连接 MySQL 服务器，创建数据库并建立数据表。

（2）使用 SELECT 查询语句查询图书信息表，如图 7.22 中❶原数据表所示。

（3）使用 UPDATE 更新语句更新数据表中数据，其代码如下：

```
update tb_mrbook set pwd='123' where name='lihui';
```







（4）当提示符窗口出现如图 7.23 所示的内容时，表示数据更新成功。

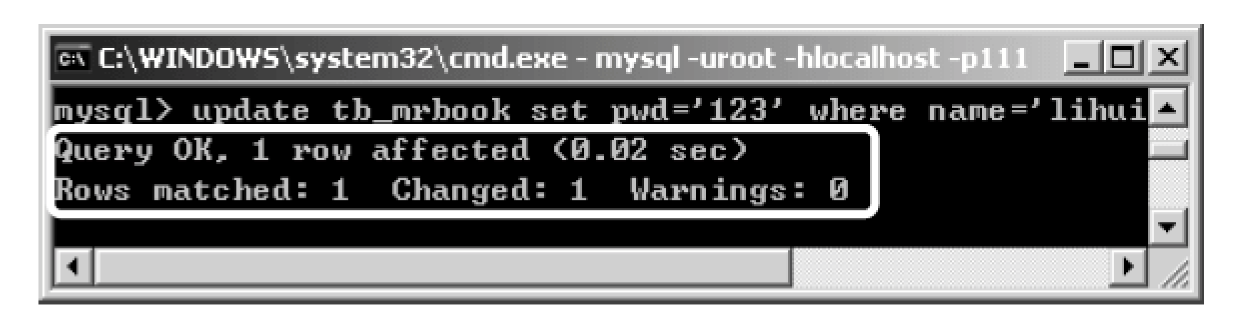


图 7.23　修改成功提示

（5）通过 SELECT 语句查询数据表中的内容，显示结果如图 7.22 中❷修改后数据所示。

（6）断开 MySQL 服务器连接，关闭 MySQL 服务。

3. 通过 DELETE 语句删除 id 等于 2 的数据

删除图书信息表中的所有数据或清空数据表的方法在实际编程中是很少见的，一般都是指定删除一条或几条数据。下面通过 DELETE 语句删除图书信息表中的指定数据，运行结果如图 7.24 所示。

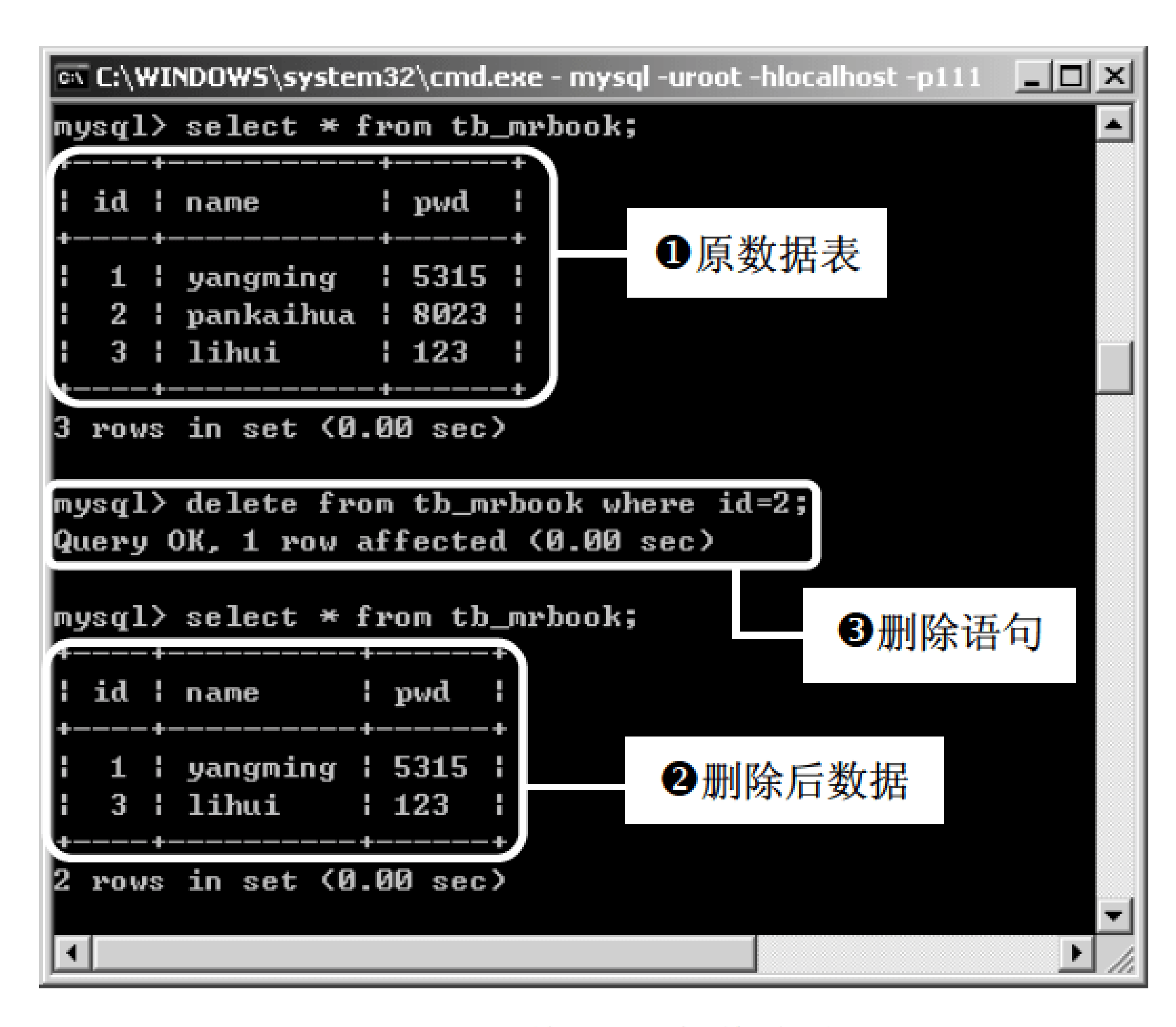


图 7.24　修改图书信息表

实现过程如下：

（1）开启 MySQL 服务，输入命令连接 MySQL，创建数据库和数据表，将数据表命名为 tb_mrbook，并通过 INSERT 语句向数据表里中添加信息，通过 SELECT 语句查看表信息，如图 7.24 中❶原数据表所示。

（2）通过 DELETE 语句删除 id 等于 2 的数据，其代码如下：

```
delete from tb_mrbook where id=2;
```

（3）当提示符窗口出现如图 7.25 所示的内容时，表示数据更新成功。

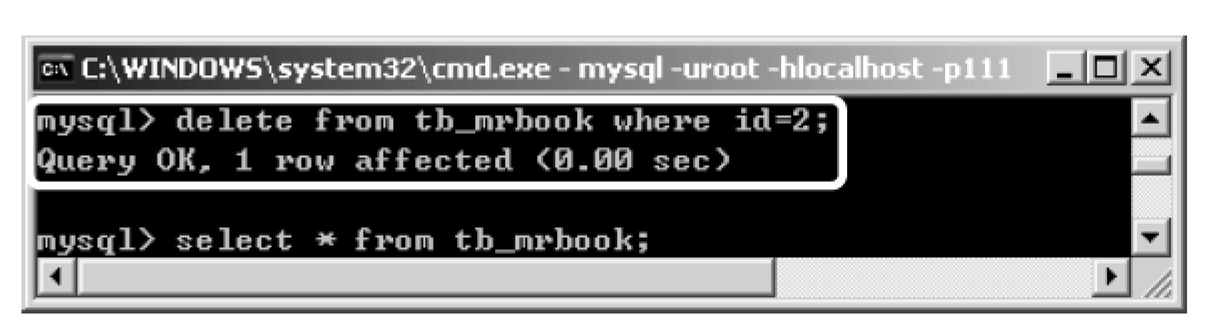
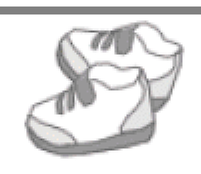
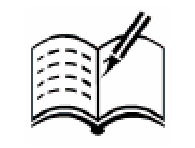


图 7.25　删除成功提示





（4）通过 SELECT 语句查询数据表中的内容，显示结果如图 7.24 中❷删除后数据所示。

（5）断开 MySQL 服务器连接，关闭 MySQL 服务。

7.6　MySQL 数据类型

在 MySQL 数据库中，每一条数据都有其数据类型。MySQL 支持的数据类型主要分成 3 类：数字类型、字符串（字符）类型、日期和时间数据类型。

7.6.1　数字类型

MySQL 支持所有的 ANSI/ISO SQL 92 数字类型。这些类型包括准确数字的数据类型（NUMERIC、DECIMAL、INTEGER 和 SMALLINT），还包括近似数字的数据类型（FLOAT、REAL 和 DOUBLE PRECISION）。其中的关键词 INT 是 INTEGER 的同义词，关键词 DEC 是 DECIMAL 的同义词。

数字类型总体可以分成整数数据类型和浮点数据类型两类，详细内容如表 7.7 和表 7.8 所示。

表 7.7　整数数据类型

数 据 类 型	取 值 范 围	说　　明	单　　位
TINYINT	符号值：-127～127 无符号值：0～255	最小的整数	1 字节
BIT	符号值：-127～127 无符号值：0～255	最小的整数	1 字节
BOOL	符号值：-127～127 无符号值：0～255	最小的整数	1 字节
SMALLINT	符号值：- 32768～32767 无符号值：0～65535	小型整数	2 字节
MEDIUMINT	符号值：- 8388608～8388607 无符号值：0～16777215	中型整数	3 字节
INT	符号值：- 2147683648～2147683647 无符号值：0～4294967295	标准整数	4 字节
BIGINT	符号值： -9223372036854775808～9223372036854775807 无符号值：0～18446744073709551615	大整数	8 字节

表 7.8　浮点数据类型

数 据 类 型	取 值 范 围	说　　明	单　　位
FLOAT	+(-)3.402823466E+38	单精度浮点数	8 或 4 字节
DOUBLE	+(-)1.7976931348623157E+308 +(-)2.2250738585072014E-308	双精度浮点数	8 字节
DECIMAL	可变	一般整数	自定义长度

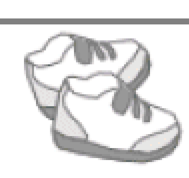
多学两招：

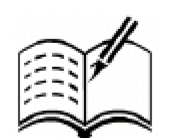
在创建表时，使用哪种数字类型应遵循以下原则：

（1）选择最小的可用类型，如果值永远不超过 127，则应使用 TINYINT，而非 INT。

（2）对于完全都是数字的数值，可以选择整数数据类型。

（3）浮点数据类型用于可能具有小数部分的数，如货物单价、网上购物交付金额等。





7.6.2 字符串类型

字符串类型可以分为3类：普通的文本字符串类型（CHAR和VARCHAR）、可变类型（TEXT和BLOB）和特殊类型（SET和ENUM）。它们的取值范围以及应用的地方都不同。

1. 普通的文本字符串类型

普通的文本字符串类型即CHAR和VARCHAR类型，CHAR列的长度被固定为创建表所声明的长度，取值在0～255之间；VARCHAR列的值是变长的字符串，取值与CHAR相同。下面介绍常规字符串类型，如表7.9所示。

表7.9 常规字符串类型

类 型	取值范围	说 明
[national] char(M) [binary|ASCII|unicode]	0～255个字符	固定长度为M的字符串，其中M的取值范围为0～255。National关键字指定了应该使用的默认字符集。Binary关键字指定了数据是否区分大小写（默认是区分大小写的）。ASCII关键字指定了在该列中使用latin1字符集。Unicode关键字指定了使用UCS字符集
char	0～255个字符	Char(M)类似
[national] varchar(M) [binary]	0～255个字符	长度可变，其他和char(M)类似

2. 可变类型

即TEXT和BLOB类型，它们的大小可以改变，TEXT类型适合存储长文本，而BLOB类型适合存储二进制数据，支持任何数据，如文本、声音和图像等。下面介绍TEXT和BLOB类型，如表7.10所示。

表7.10 TEXT和BLOB类型

类 型	最大长度（字节数）	说 明
TINYBLOB	2^8-1（225）	小BLOB字段
TINYTEXT	2^8-1（225）	小TEXT字段
BLOB	2^16-1（65535）	常规BLOB字段
TEXT	2^16-1（65535）	常规TEXT字段
MEDIUMBLOB	2^24-1（16777215）	中型BLOB字段
MEDIUMTEXT	2^24-1（16777215）	中型TEXT字段
LONGBLOB	2^32-1（4294967295）	长BLOB字段
LONGTEXT	2^32-1（4294967295）	长TEXT字段

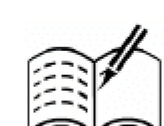
3. 特殊类型

特殊类型SET和ENUM的介绍如表7.11所示。

表7.11 SET和ENUM类型

类 型	最 大 值	说 明
Enum ("value1", "value2", …)	65535	该类型的列只可以容纳所列值之一或为NULL
Set ("value1", "value2", …)	64	该类型的列可以容纳一组值或为NULL





多学两招：

在创建表时，使用字符串类型的过程中应遵循以下原则：

（1）从速度方面考虑，要选择固定的列，可以使用 CHAR 类型。

（2）要节省空间，使用动态的列，可以使用 VARCHAR 类型。

（3）要将列中的内容限制在一种选择，可以使用 ENUM 类型。

（4）如果允许在一个列中有多于一个的条目，可以使用 SET 类型。

（5）如果要搜索的内容不区分大小写，可以使用 TEXT 类型。

（6）如果要搜索的内容区分大小写，可以使用 BLOB 类型。

7.6.3 日期和时间数据类型

日期和时间数据类型包括 DATETIME、DATE、TIMESTAMP、TIME 和 YEAR。其中的每种类型都有其取值的范围，如果赋予它一个不合法的值，将会被“0”代替。下面介绍日期和时间数据类型，如表 7.12 所示。

表 7.12 日期和时间数据类型

类　型	取 值 范 围	说　明
DATE	1000-01-01　9999-12-31	日期，格式 YYYY-MM-DD
TIME	-838:58:59　835:59:59	时间，格式 HH:MM:SS
DATETIME	1000-01-01 00:00:00 9999-12-31 23:59:59	日期和时间，格式 YYYY-MM-DD HH:MM:SS
TIMESTAMP	1970-01-01 00:00:00 2037 年的某个时间	时间标签，在处理报告时使用显示格式取决于 M 的值
YEAR	1901-2155	年份可指定两位数字和 4 位数字的格式

在 MySQL 中，日期的顺序是按照标准的 ANSISQL 格式进行输出的。

7.7 phpMyAdmin 管理 MySQL 数据库

视频讲解：配套资源\mr\7\video\phpMyAdmin 管理 MySQL 数据库.exe

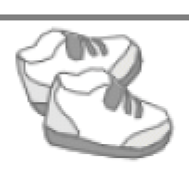
phpMyAdmin 是众多 MySQL 图形化管理工具中应用最广泛的一种，是一款使用 PHP 开发的 B/S 模式的 MySQL 客户端软件，该工具是基于 Web 跨平台的管理程序，并且支持简体中文。用户可以在其官方网站 www.phpMyAdmin.net 上免费下载到最新的版本。phpMyAdmin 为 Web 开发人员提供了类似于 Access、SQL Server 的图形化数据库操作界面，通过该管理工具可以完全对 MySQL 进行操作，如创建数据库、数据表以及生成 MySQL 数据库脚本文件等。

7.7.1 管理数据库

在浏览器地址栏中输入 http://localhost/phpMyAdmin/，进入 phpMyAdmin 图形化管理主界面，接下来就可以进行 MySQL 数据库的操作了。下面分别介绍如何创建、修改和删除数据库。

1. 创建数据库

打开 phpMyAdmin 的主界面，首先在文本框中输入数据库的名称“db_study”，然后在下拉





列表框中选择所要使用的编码，一般选择“gb2312_Chinese_ci”简体中文编码格式，再单击“创建”按钮，创建数据库，如图 7.26 所示。成功创建数据库后，将显示如图 7.27 所示的界面。

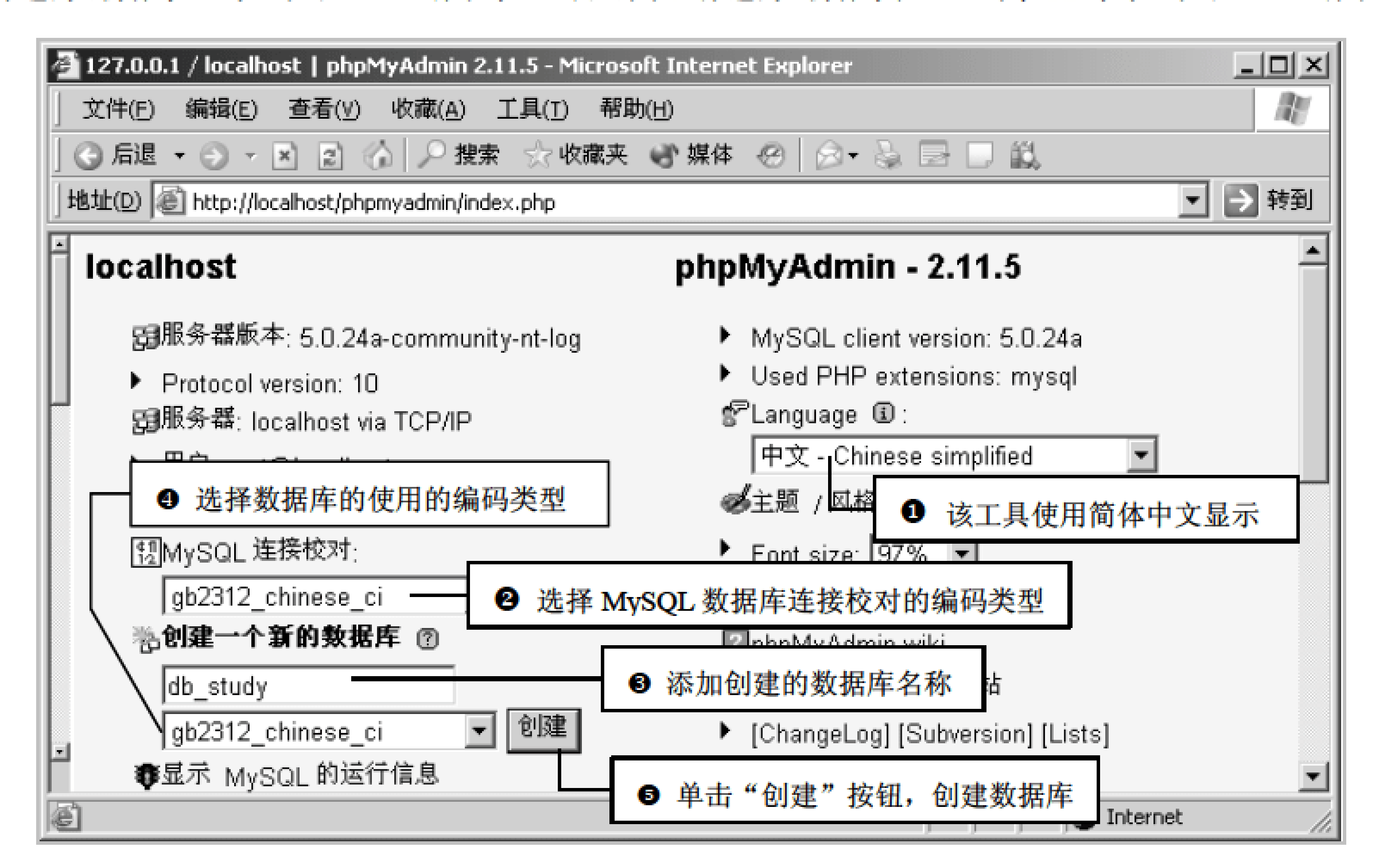


图 7.26　phpMyAdmin 的主界面

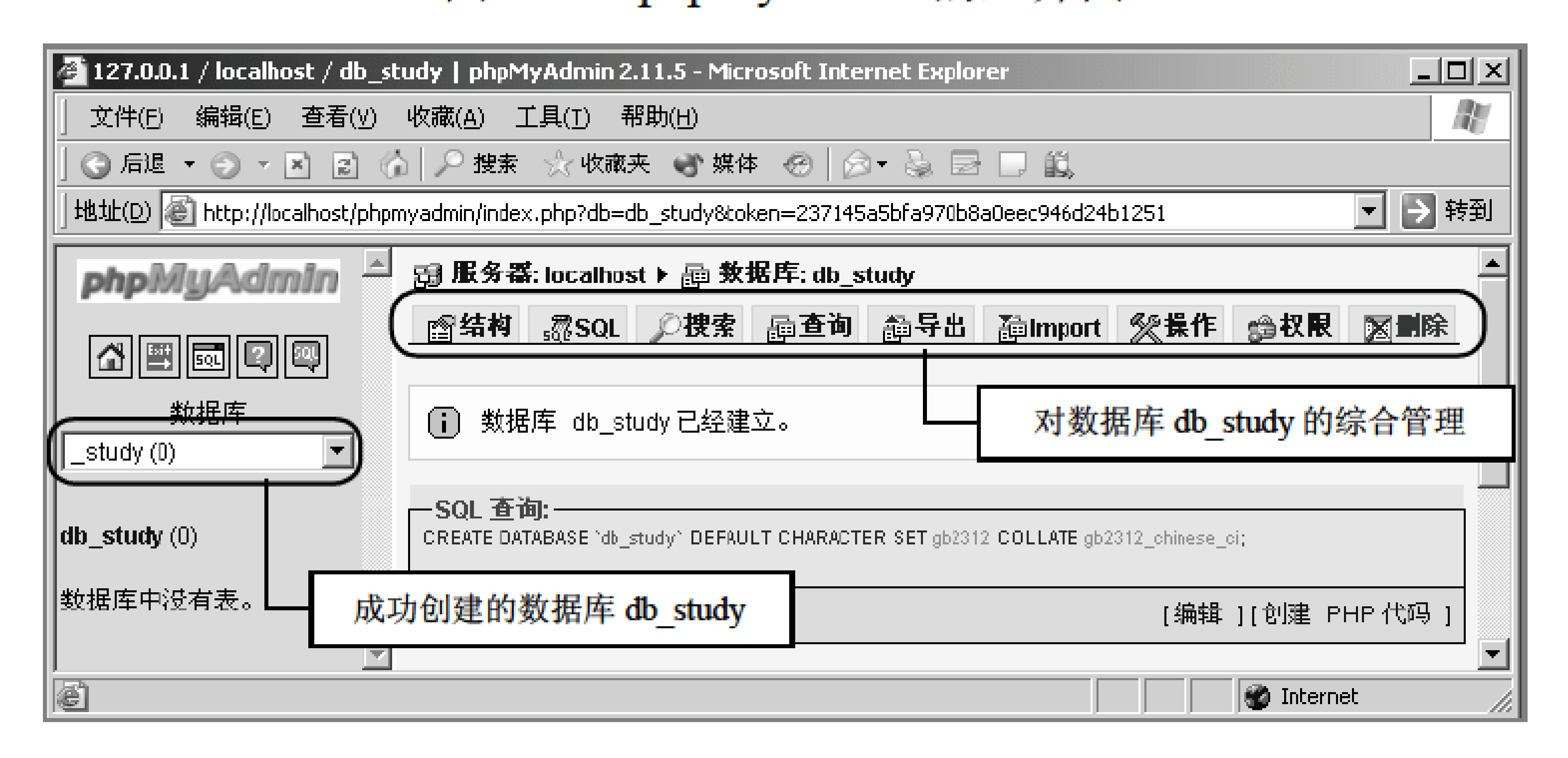


图 7.27　成功创建数据库

多学两招：

在如图 7.27 所示界面的右侧区域中，可以对该数据库进行相关操作，如结构、SQL、导出、搜索、查询、删除等，单击相应的超链接可进入相应的操作界面。但是在创建数据库后还没有创建数据表的情况下，只能够执行结构、SQL、Import、操作、权限和删除 6 项操作，其他 3 项操作 搜索 查询 导出 不能执行，当光标指向其超链接时，弹出不可用标记☒。

2．修改数据库

在如图 7.27 所示界面的右侧区域中，可以对当前数据库进行修改，单击 操作 超链接，即可进入修改操作界面。

在该界面中，可以对当前数据库执行创建数据表的操作，只要在创建数据表的提示信息下面的两个文本框中分别输入要创建的数据表的名称和字段总数，然后单击“执行”按钮即可进入到创建数据表结构界面。

也可以对当前的数据库重命名，在“重新命名数据库为”文本框中输入新的数据库名称，然后单击“执行”按钮，即可成功修改数据库名称，如图 7.28 所示。

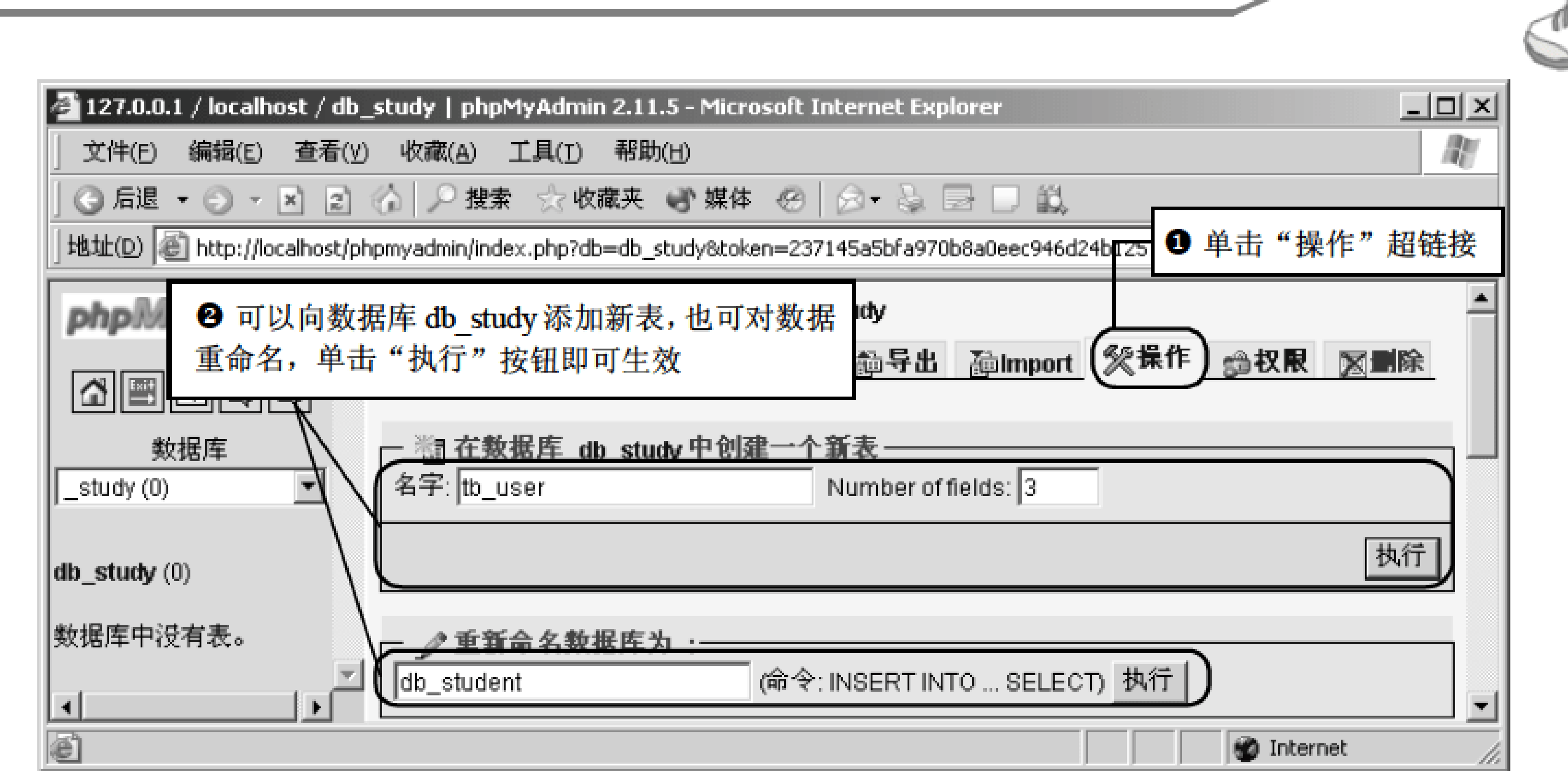


图 7.28　修改数据库



3. 删除数据库

要删除某个数据库，首先在左侧的下拉菜单中选择该数据库，然后单击界面右侧区域中的删除超链接（如图 7.28 所示）即可成功删除指定的数据库。

7.7.2　管理数据表

管理数据表是以选择指定的数据库为前题，然后在该数据库中创建并管理数据表。下面就来介绍如何创建、修改和删除数据表。

1. 创建数据表

创建数据库 db_study 后，在右侧的操作页面中输入数据表的名称和字段数，然后单击“执行”按钮，即可创建数据表，如图 7.29 所示。

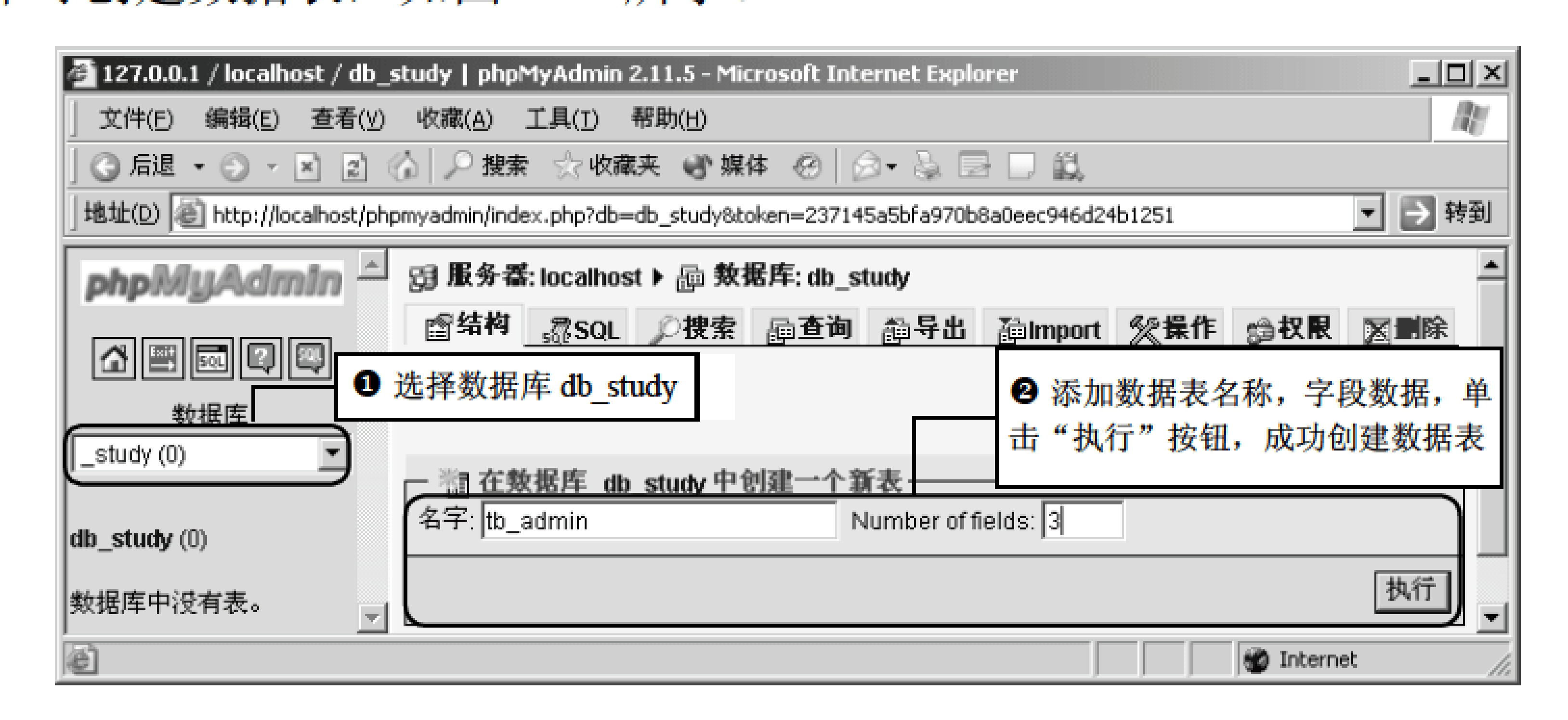


图 7.29　创建数据表

成功创建数据表 tb_admin 后，将显示数据表结构界面。在表单中输入各个字段的详细信息，包括字段名、数据类型、长度/值、编码格式、是否为空、主键等，以完成对表结构的详细设置。当所有的信息都输入完成以后，单击“保存”按钮，创建数据表结构，如图 7.30 所示。成功创建数据表结构后，将显示如图 7.31 所示的界面。

多学两招：

单击“执行”按钮，可以对数据表结构以横版显示进行表结构编辑。





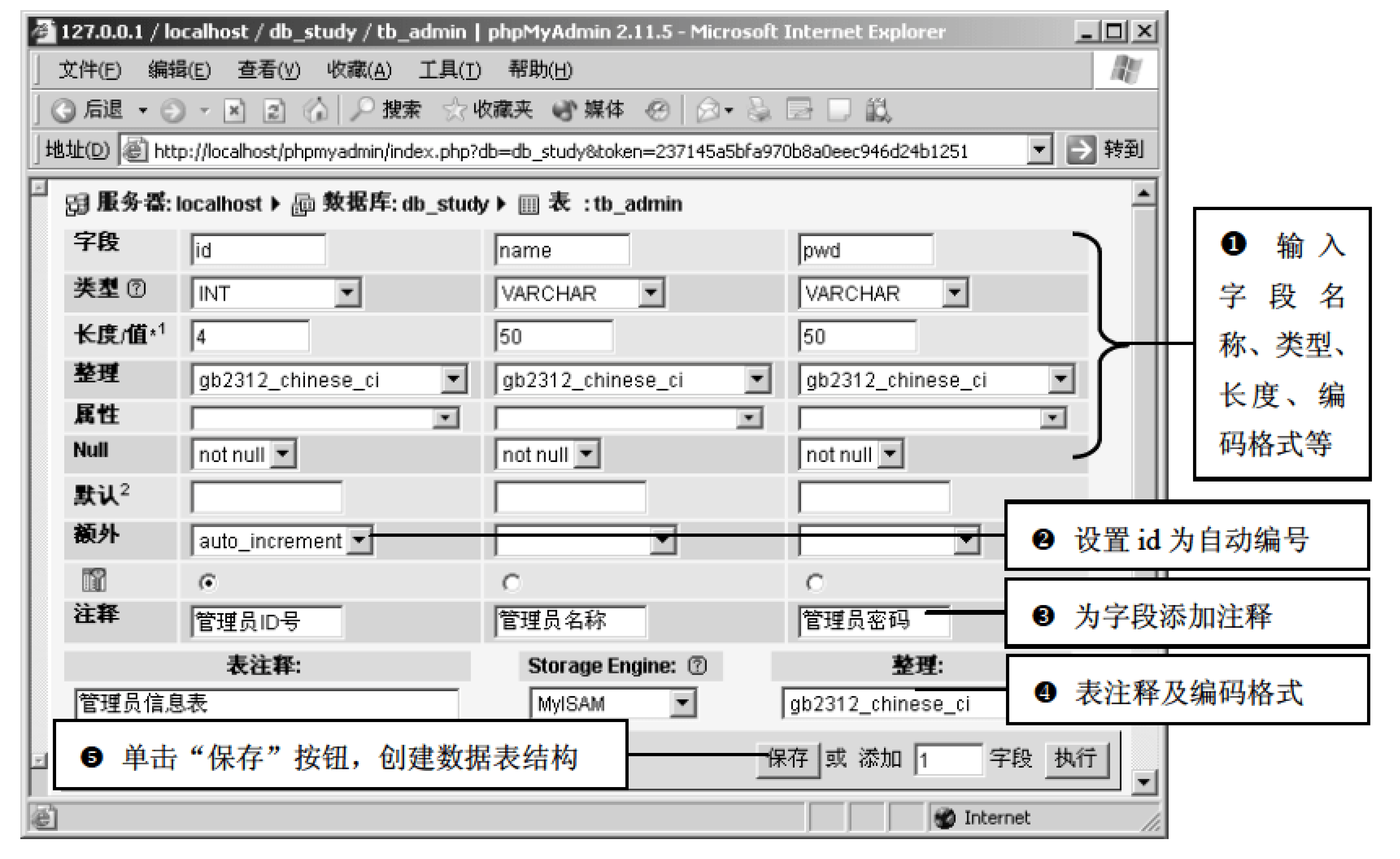


图 7.30　创建数据表结构



图 7.31　成功创建数据表结构

2. 修改数据表

一个新的数据表被创建后，进入到数据表页面中，在这里可以通过改变数据表的结构来对数据表进行修改，可以执行添加新的列、删除列、索引列、修改列的数据类型或字段的长度/值等操作，如图 7.32 所示。

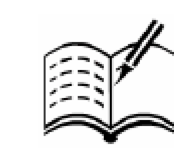


图 7.32　修改数据表结构

3. 删除数据表

要删除某个数据表，首先在左侧的下拉菜单中选择该数据表所在数据库，在指定的数据库中选择要删除的数据表，然后单击右侧界面中的删除超链接（如图 7.32 所示），即可成功删除指定的数据表。





Note

7.7.3　管理数据记录

单击 phpMyAdmin 主界面中的 SQL 超链接，打开 SQL 语句编辑区。在该区域输入完整的 SQL 语句，可实现数据的查询、添加、修改和删除操作。

1. 使用 SQL 语句插入数据

在 SQL 语句编辑区应用 insert 语句向数据表 tb_admin 中插入数据后，单击“执行”按钮，可向数据表中插入一条数据，如图 7.33 所示。如果提交的 SQL 语句有错误，则系统会给出一个警告，提示用户修改；如果提交的 SQL 语句正确，则弹出如图 7.34 所示的提示信息。

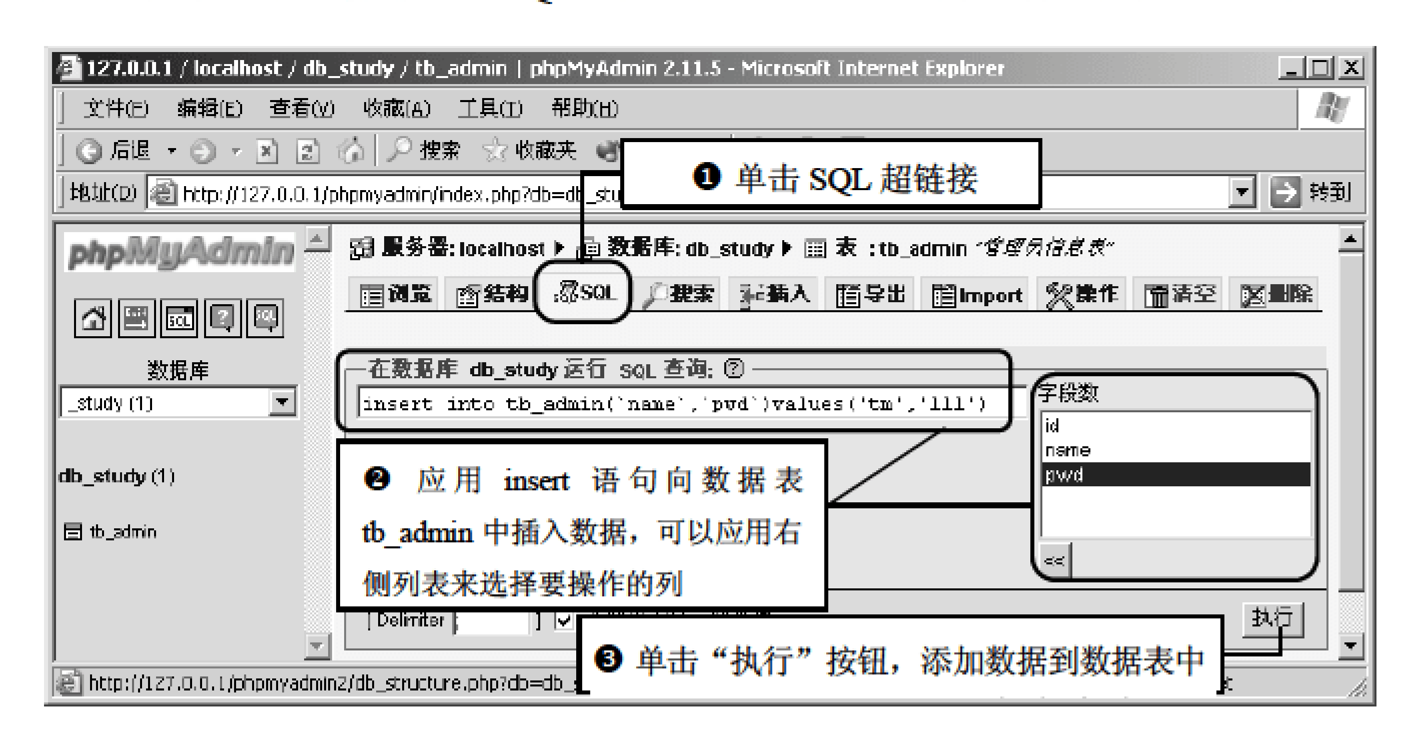


图 7.33　使用 SQL 语句向数据表中插入数据



图 7.34　成功添加数据信息

多学两招：

为了编写方便，可以利用图 7-33 右侧的属性列表来选择要操作的列，只要选中要添加的列，双击其选项或单击“<<”按钮即可添加列名称。

2. 使用 SQL 语句修改数据

在 SQL 语句编辑区应用 update 语句修改数据信息，将 ID 为 1 的管理员的名称改为“纯净水”，密码改为“111”，添加的 SQL 语句如图 7.35 所示。

单击“执行”按钮，数据修改成功。比较修改前后的数据如图 7.36 所示。

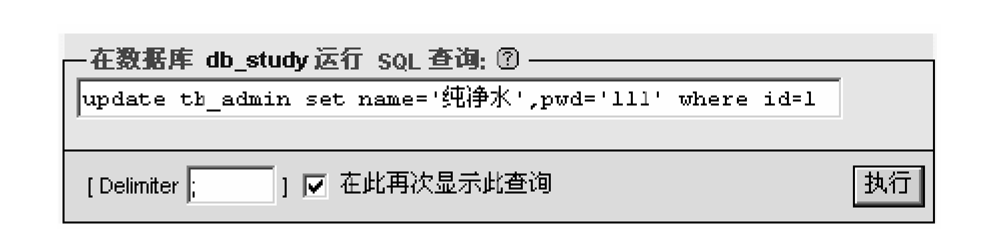


图 7.35　添加修改数据信息的 SQL 语句

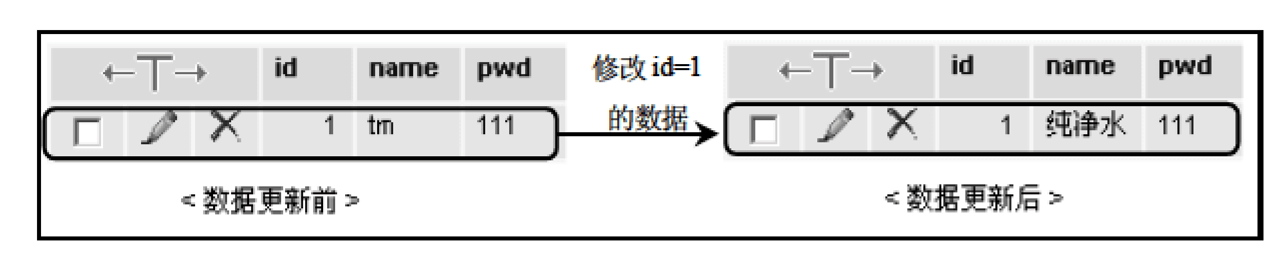


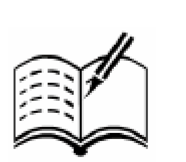
图 7.36　修改单条数据的实现过程

3. 使用 SQL 语句查询数据

在 SQL 语句编辑区应用 select 语句检索指定条件的数据信息，将 ID 小于 4 的管理员全部显示出来，添加的 SQL 语句如图 7.37 所示。

单击“执行”按钮，该语句的实现过程如图 7.38 所示。



Note

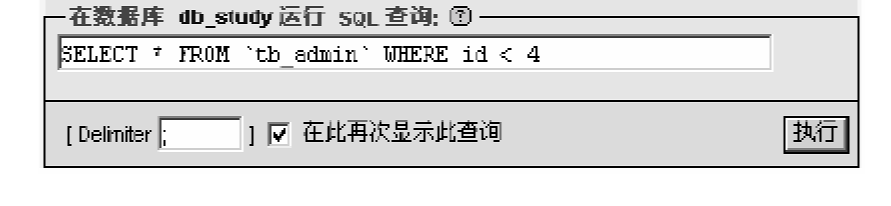


图 7.37　添加查询数据信息的 SQL 语句

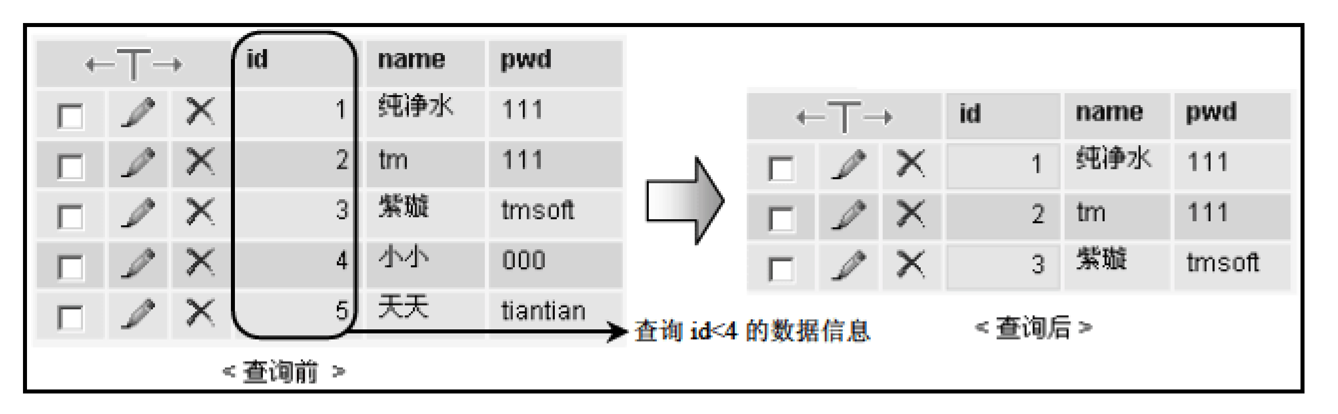


图 7.38　查询指定条件的数据信息的实现过程

除了可以对整个表进行简单查询外，还可以执行复杂的条件查询（使用 where 子句提交 LIKE、ORDER BY、GROUP BY 等条件查询语句）及多表查询，读者可通过上机进行实践，灵活运用 SQL 语句功能。

4. 使用 SQL 语句删除数据

在 SQL 语句编辑区应用 delete 语句检索指定条件的数据或全部数据信息，删除名称为“tm”的管理员信息，添加的 SQL 语句如图 7.39 所示。

脚下留神：

如果 Delete 语句后面没有 Where 条件值，那么将删除指定数据表中的全部数据。

单击“执行”按钮，弹出确认删除操作对话框，单击“确定”按钮，执行数据表中指定条件的删除操作。该语句的实现过程如图 7.40 所示。

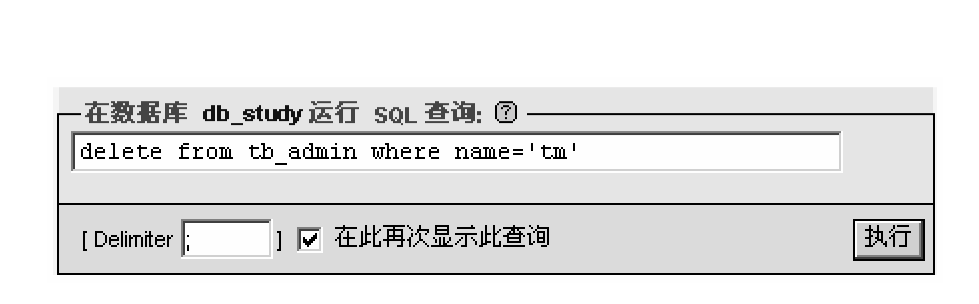


图 7.39　添加删除指定数据信息的 SQL 语句

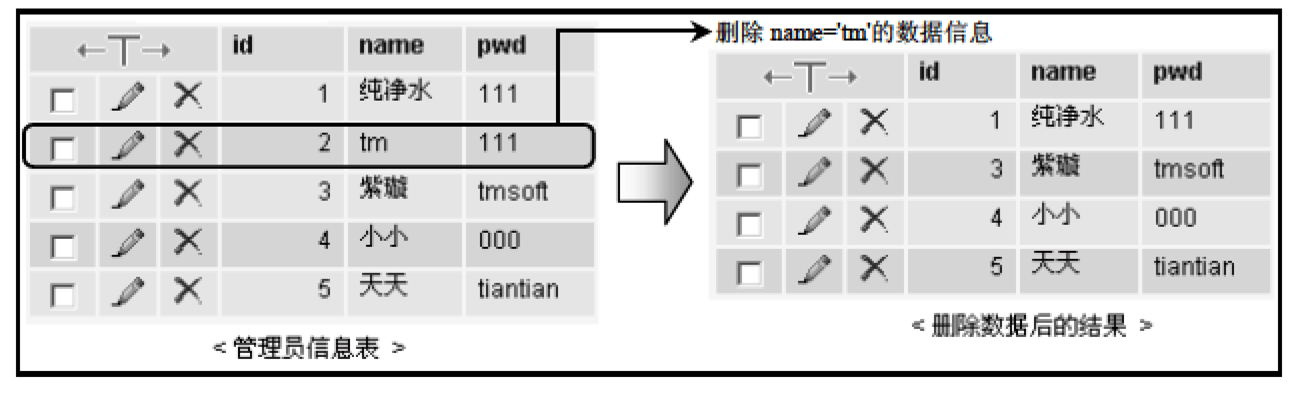


图 7.40　删除指定条件的数据信息的实现过程

5. 通过 form 表单插入数据

选择某个数据表后，单击插入超链接，可进入插入数据界面，如图 7.41 所示。在界面中输入各字段值，单击“执行”按钮即可插入记录。默认情况下，一次可以插入两条记录。

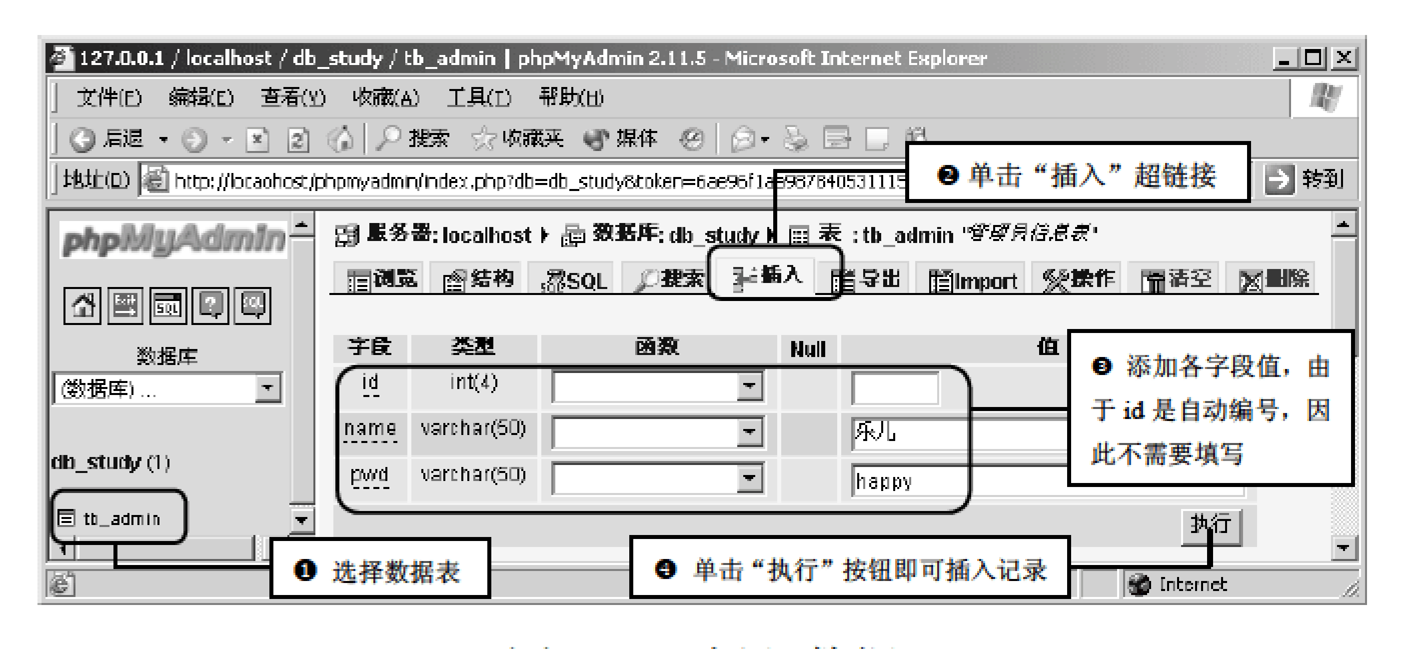
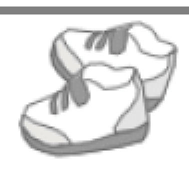


图 7.41　插入数据

6. 浏览数据

选择某个数据表后，单击浏览超链接，可进入浏览数据界面，如图 7.42 所示。单击每行记录中的按钮，可以对该记录进行编辑；单击每行记录中的按钮，可以删除该条记录。





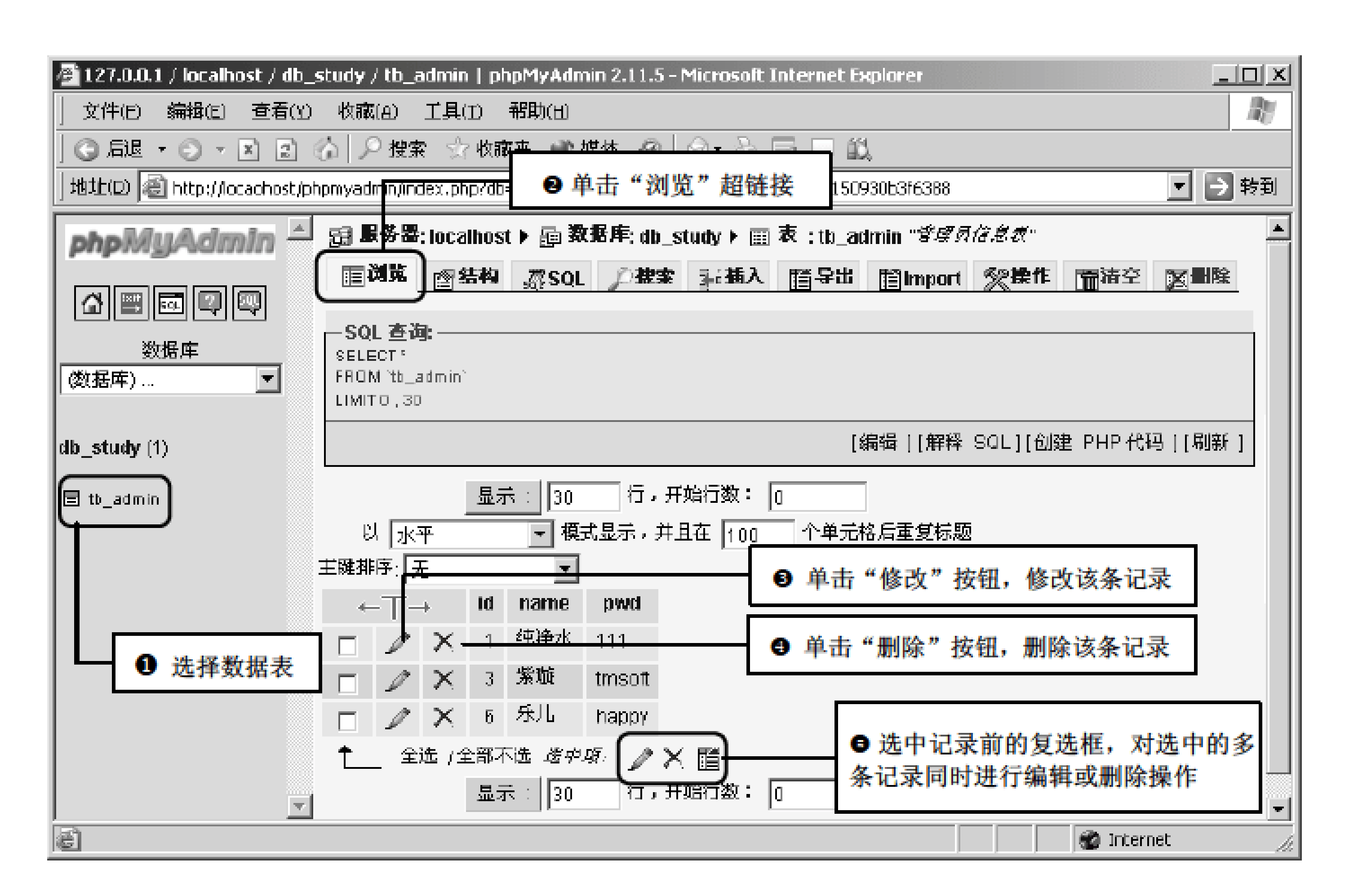


图 7.42　浏览数据

7．搜索数据

选择某个数据表后，单击“搜索”超链接，可进入搜索页面，如图 7.43 所示。在该页面中，可以在选择字段的列表框中选择一个或多个列，如果要选择多个列，在按下 Ctrl 键的同时单击要选择的字段名即可，查询结果将按照选择的字段名进行输出。

在该界面中可以对记录按条件进行查询。查询方式有两种：第一种方式是选择构建 where 语句查询，即直接在“添加搜索条件（‘where’ 语句的主体）”文本框中输入查询语句，然后单击“执行”按钮；第二种方式是使用“按例查询”，即选择查询的条件，并在文本框中输入要查询的值，然后单击“执行”按钮。

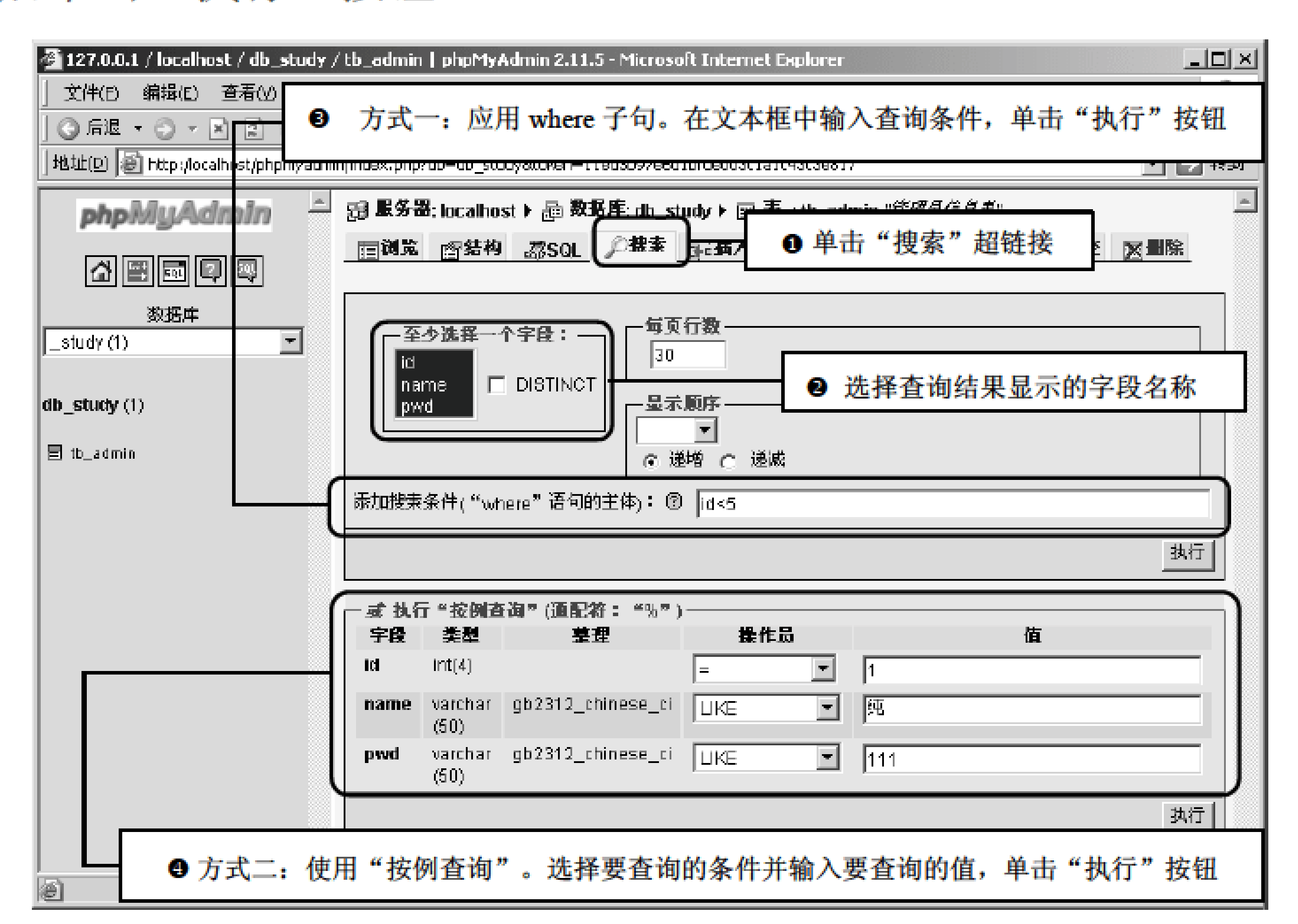
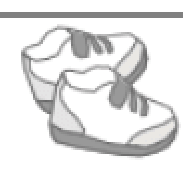
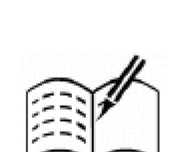


图 7.43　搜索查询

7.7.4　导入和导出数据

导入和导出 MySQL 数据库脚本是互逆的两个操作。导入是执行扩展名为.sql 的文件，将数据导入到数据库中；导出是将数据表结构、表记录存储为扩展名为.sql 的脚本文件。通过导入和导出操作，可实现数据库的备份和还原。







1. 导出 MySQL 数据库脚本

单击 phpMyAdmin 主界面中的导出超链接，打开导出编辑区，如图 7.44 所示。选择欲生成脚本文件的数据表，选择导出文件的格式，这里默认使用选项“SQL”，选中“另存为文件”复选框，单击“执行”按钮，弹出如图 7.45 所示的“文件下载”对话框，单击“保存”按钮，即可将脚本文件以.sql 格式存储在指定位置。

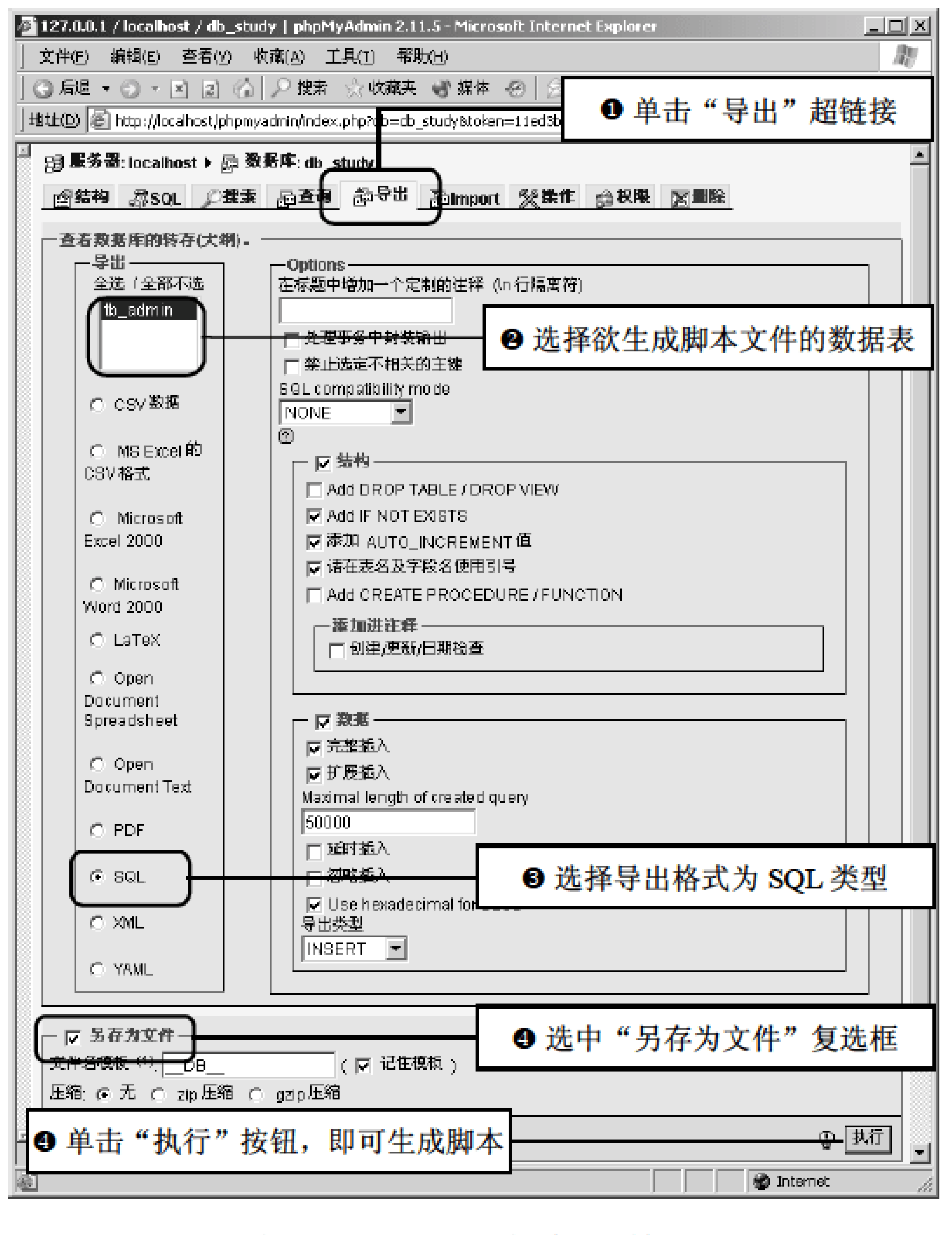


图 7.44　生成 MySQL 脚本文件设置界面

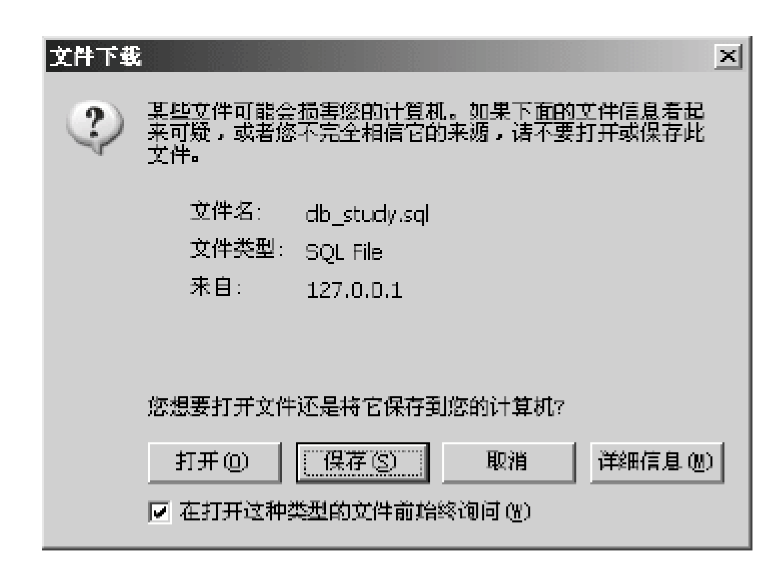


图 7.45　“文件下载”对话框

2. 导入 MySQL 数据库脚本

单击Import超级链接，进入执行 MySQL 数据库脚本界面，单击“浏览”按钮查找脚本文件（如 db_study.sql）所在的位置，如图 7.46 所示，选中 SQL 单选按钮，单击“执行”按钮，即可执行 MySQL 数据库脚本文件。

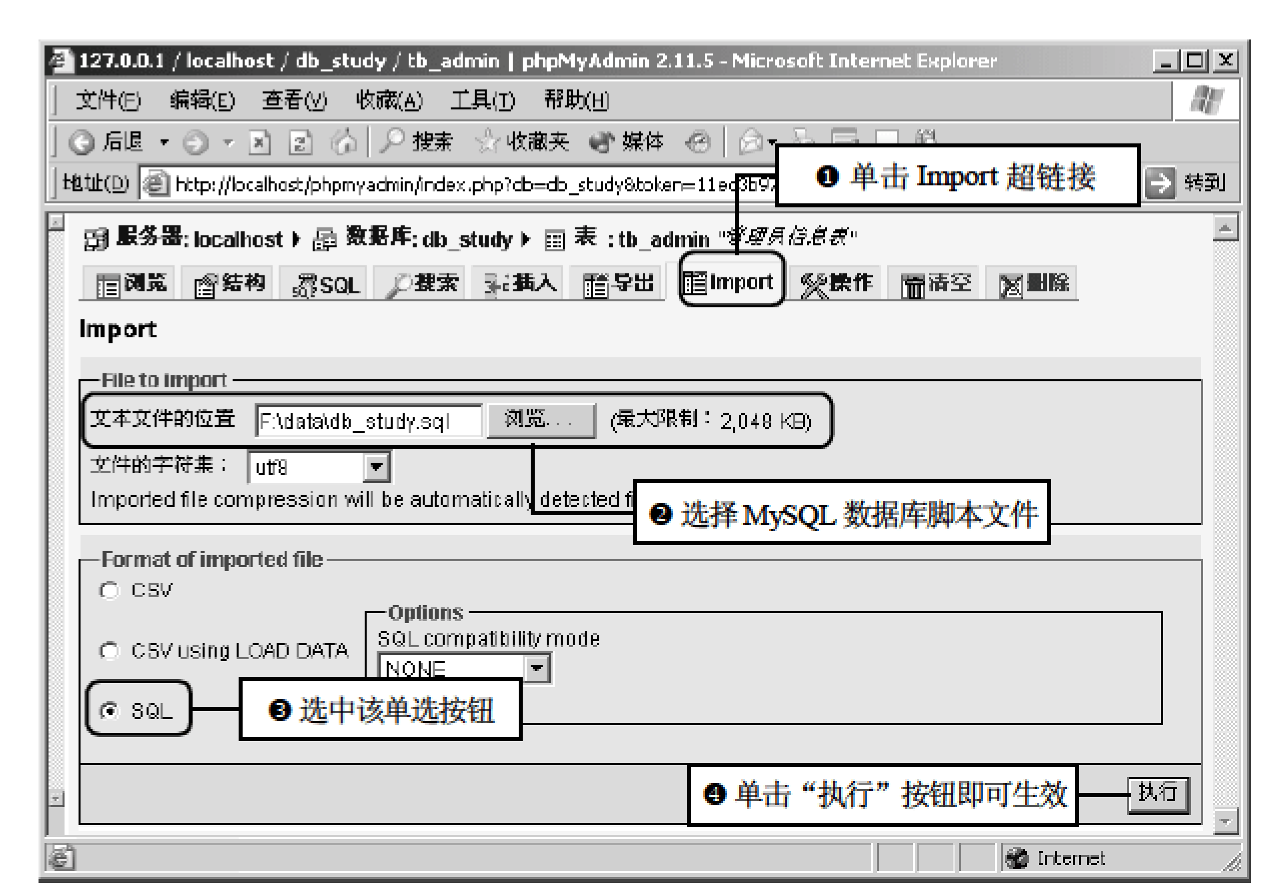
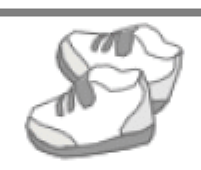


图 7.46　执行 MySQL 数据库脚本文件





脚下留神：

在执行 MySQL 脚本文件前，首先应检测是否有与所导入数据库脚本文件同名的数据库，如果没有，则要在数据库中创建一个与数据文件中的数据库同名的数据库，然后再执行 MySQL 数据库脚本文件。另外，在当前数据库中，不能有与将要导入的数据表重名的数据表存在，如果有，则导入文件将会失败，提示错误信息。

多学两招：

也可以通过单击 phpMyAdmin 图形化工具左侧的SQL按钮，在打开的对话框中单击“导入文件”超链接，然后选择脚本文件所在的位置，来执行脚本文件。

☎小测试

1. 通过 phpMyAdmin 添加服务器新用户

在 phpMyAdmin 图形化管理工具中，不但可以对 MySQL 数据库进行各种操作，而且可以添加服务器的新用户，并对新添加的用户设置权限。

在 phpMyAdmin 中添加 MySQL 服务器新用户的步骤如下：

（1）单击 phpMyAdmin 主界面中的权限超链接，打开服务器用户操作界面，如图 7.47 所示。

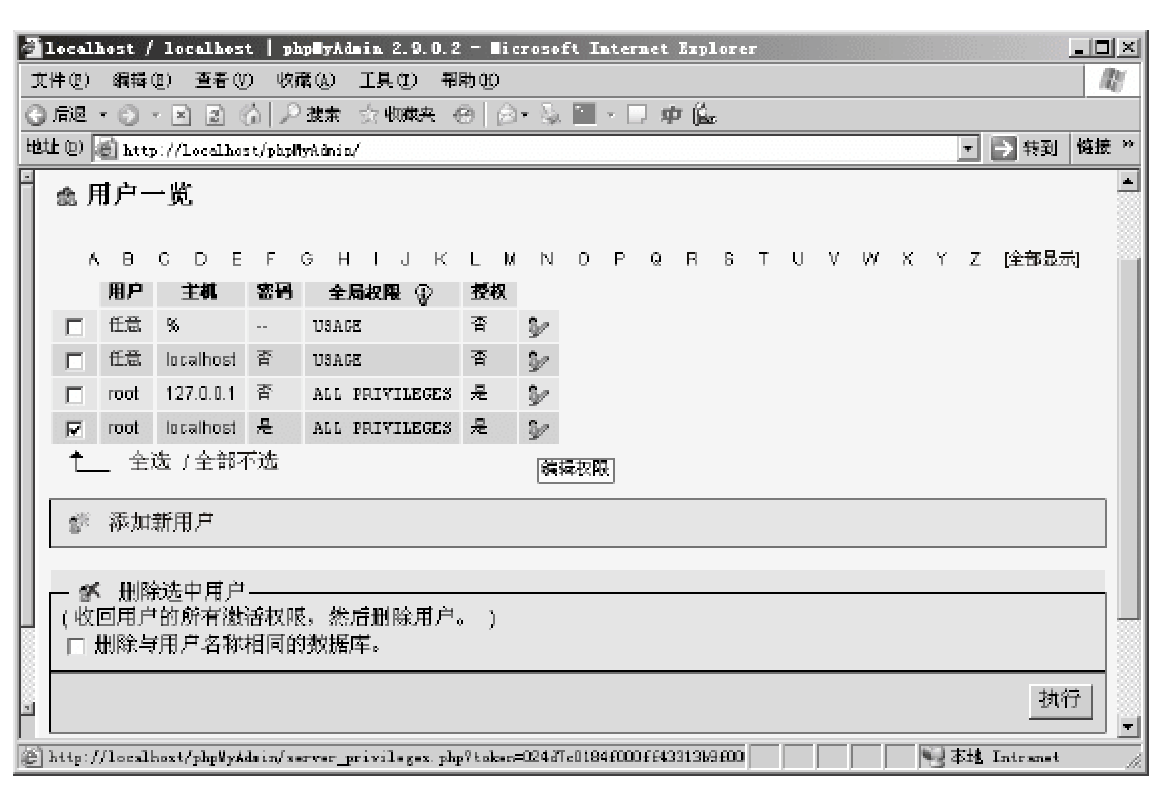


图 7.47　服务器用户一览表

（2）在该界面中单击“添加新用户”按钮。打开如图 7.48 所示的界面，设置用户名、密码、主机，并对新用户的权限进行设置。设置完成后，单击“执行”按钮，完成对新用户的添加操作，返回主页面，将提示新用户添加成功。

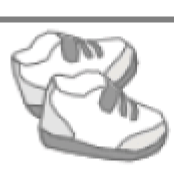
2. 在 phpMyAdmin 中重置 MySQL 服务器登录密码

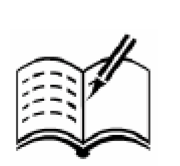
在 phpMyAdmin 图形化管理工具中，可以重置 MySQL 服务器的登录密码。其步骤如下：

（1）单击 phpMyAdmin 主界面中的权限超链接，打开服务器用户操作界面，如图 7.49 所示。

（2）在该界面中可以对指定用户的权限进行编辑，同时可以添加新用户和删除指定的用户。这里选择指定的用户，单击（编辑权限）超链接，对指定用户的权限进行设置，进入到如图 7.50 所示的界面。

（3）在如图 7.50 所示的界面中，可以设置用户的权限、修改密码、更改登录用户信息和复制用户。在输入新密码和确认密码之后，单击“执行”按钮，完成对用户密码的修改操作，





返回主页面，将提示密码修改成功。

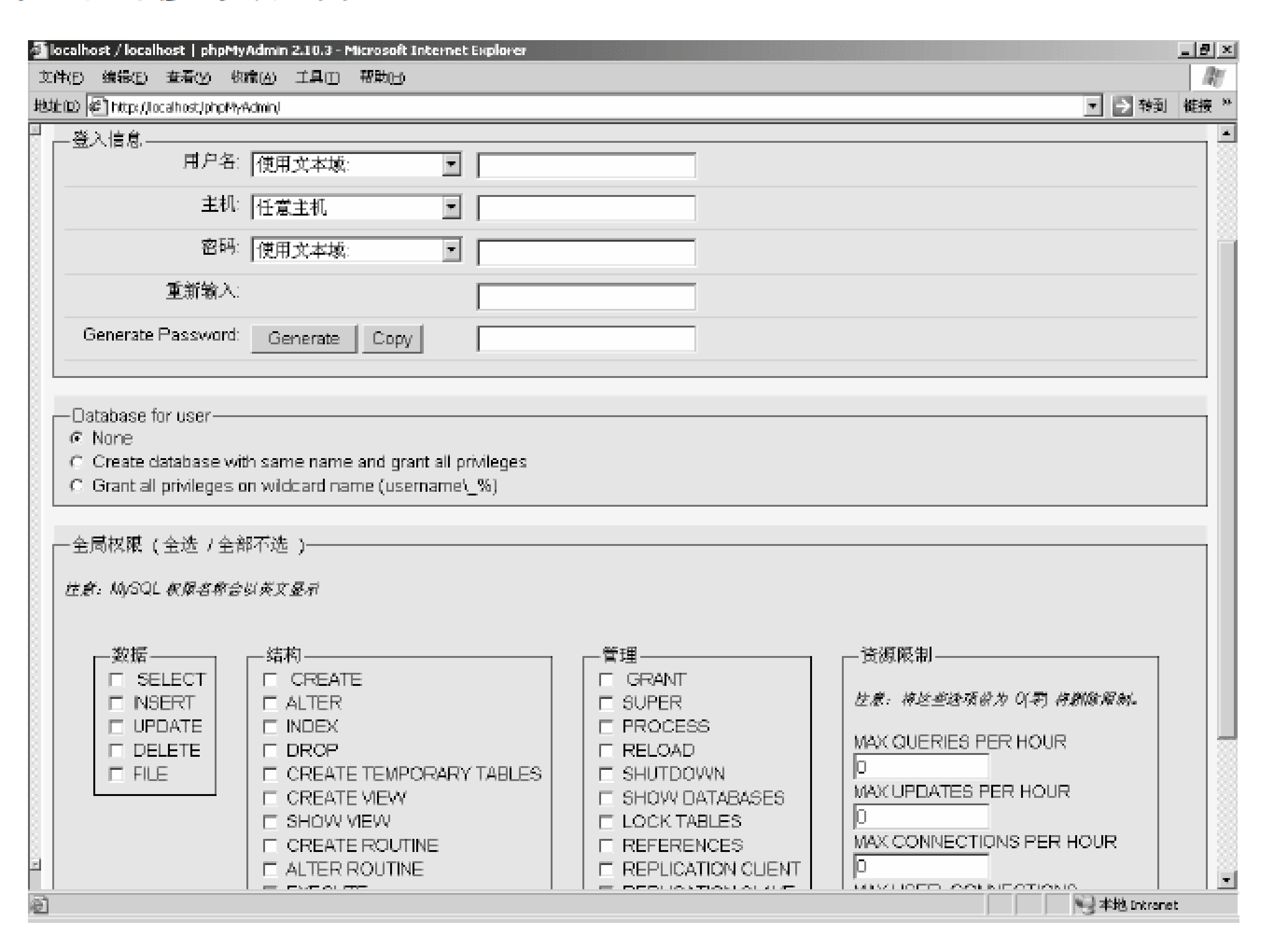


图 7.48　设置添加用户信息

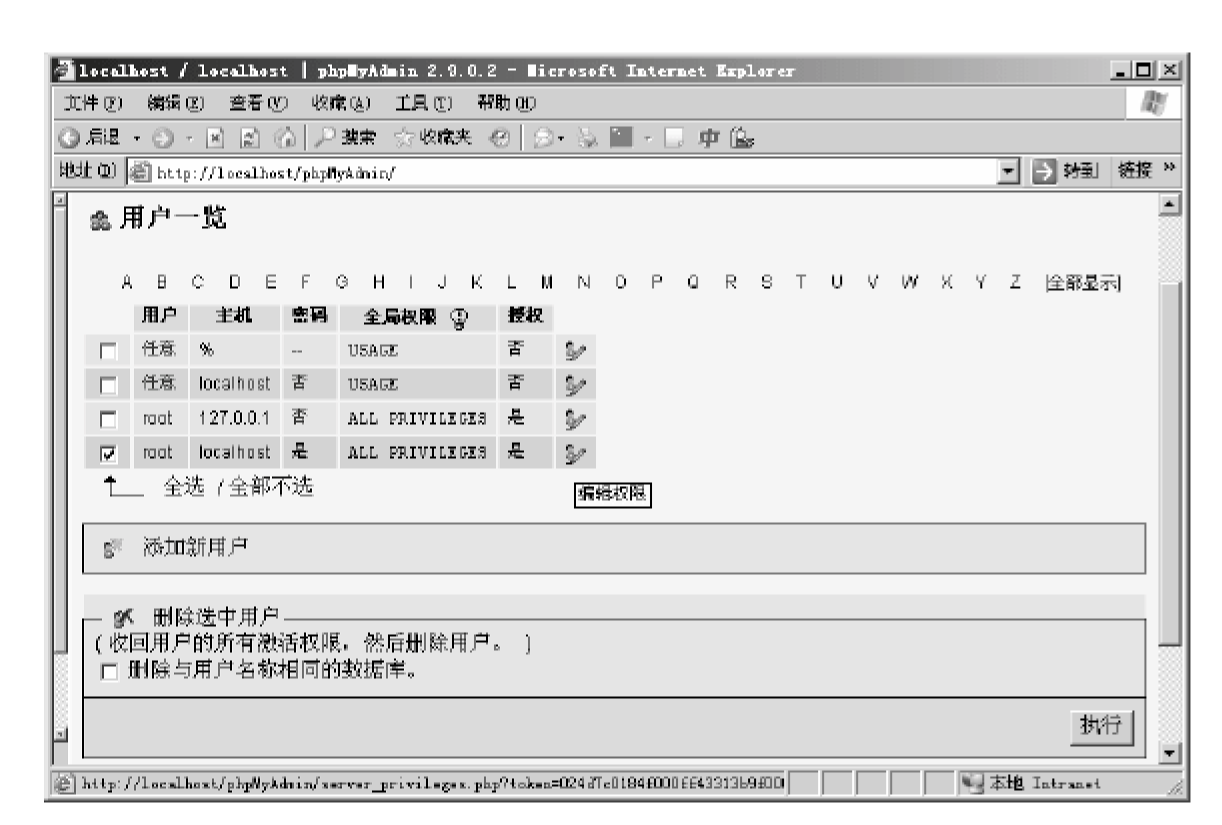


图 7.49　服务器用户一览表

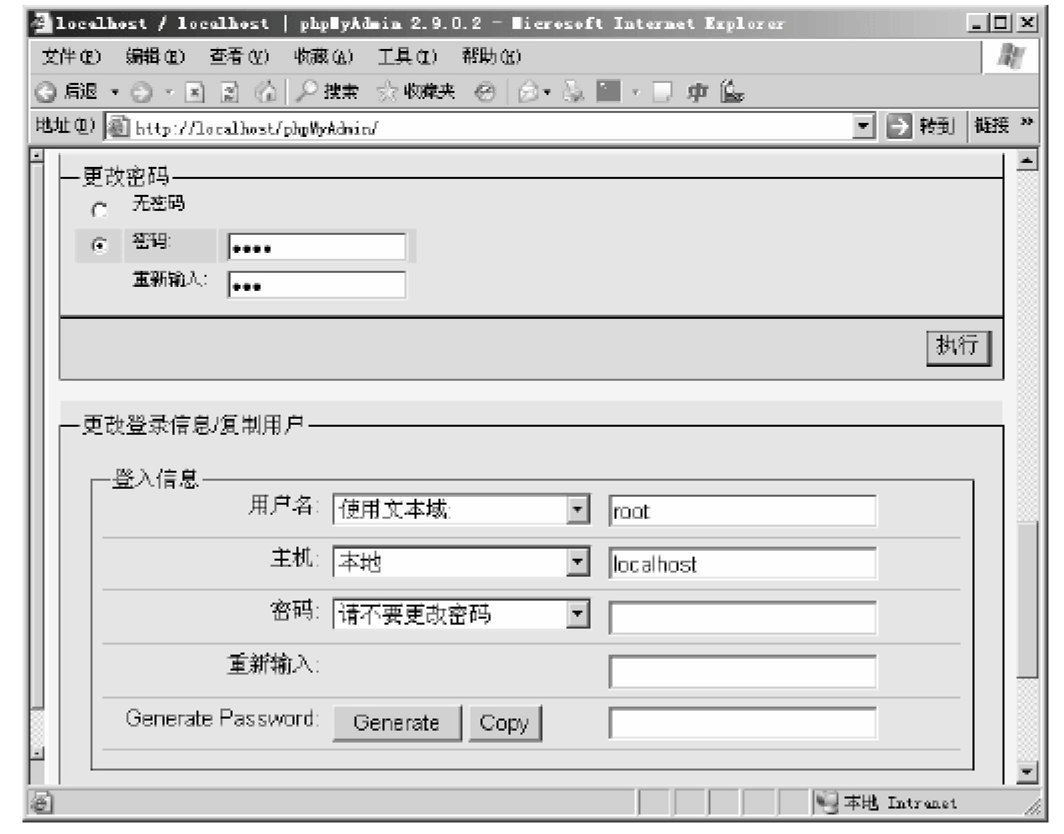


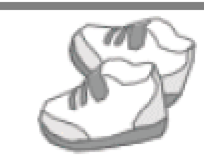
图 7.50　编辑用户权限

本 章 摘 要

1. MySQL 服务器的启动和停止方法。
2. MySQL 数据库的创建、选择和删除操作。
3. MySQL 数据表的创建、修改、重命名和删除操作。
4. MySQL 数据的添加、修改和删除操作。
5. MySQL 数据的查询操作，其中对 where 子句进行了系统的讲解。
6. MySQL 数据的类型。
7. MySQL 数据库图形化管理工具 phpMyAdmin 的使用方法。

习　　题

1. 关于 MySQL_select_db 的作用描述正确的是（　　）。



A．连接数据库　　B．连接并选取数据库
C．连接并打开数据库　　D．选取数据库

2．MySQL_insert_id()函数的作用是（　　）。
A．查看下一次插入记录时的 ID 号　　B．查看刚刚插入过的自动增长 ID 值
C．查看一共做过多少次 insert 操作　　D．查看一共有多少条记录

3．修改 MySQL 用户 root 的密码的指令是（　　）。
A．MySQLadmin –u root password test　　B．MySQL –u root password test
C．MySQL –u root –p test　　D．MySQL –u root –password test

4．查询所有年龄在 20 岁以下的学生姓名及其年龄，以下正确的是（　　）。
A．SELECT Sname，Sage FROM Student WHERE Sage <=20;
B．SELECT Sname，Sage FROM Student WHERE NOT Sage <20;
C．SELECT Sname，Sage FROM Student WHERE NOT Sage >20;
D．SELECT Sname，Sage FROM Student WHERE Sage <20;

5．导出数据库正确的方法为（　　）。
A．MySQLdump 数据库名 >文件名　　B．MySQLdump 数据库名 >>文件名
C．MySQLdump 数据库名 文件名　　D．MySQLdump 数据库名 =文件名

6．删除数据表中 id=5 的记录，其用到的主要代码是（　　）。

7．通过命令模式设置编码格式的函数是（　　）。

8．MySQL 数据库的备份和恢复，可以通过数据库的（　　）和（　　）来完成。

9．假设有一个数据库 mydb，其中有一个表 tb_mrbook，查询 tb_mrbook 表，按照图书价格降序排列，显示 3 条记录，其代码是（　　）。

10．假设有一个数据库 mydb，其中有一个表 tb1，表中有 6 个字段，主键为 ID，有 10 条记录，ID 从 0 到 9，以下代码输出的结果是（　　）。

```
select  *  from  tb1 where Id<=5
```



实战模拟

实战模拟 1　手动备份、恢复 MySQL 数据库

本实例介绍一种最为简单的数据库备份、恢复方式，也是使用 MySQL 数据库最常用的备份、恢复方式，即通过手动方式备份、恢复数据库。

通过手动方式备份整个数据库文件，操作步骤如下：

（1）找到 MySQL 数据库的安装路径，在 MySQL 根目录下找到 data 文件夹。本书中 data 文件夹的存储位置是 F:\xampp\xampp\mysql\data。

（2）打开 data 文件夹，选择要备份的数据库名称。

（3）将要备份的数据库文件夹复制到指定的位置下。

通过手动方式恢复整个数据库文件，操作步骤如下：

（1）找到通过手动方式备份的数据库文件。

（2）将数据库文件复制到 MySQL 数据库安装路径下的 data 文件夹中，这样即可完成通过手动方式恢复数据库。

第8章

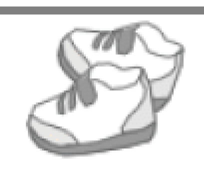
PHP 数据库编程技术

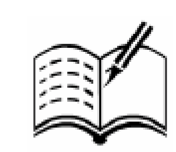
（ 自学视频、源程序：配套资源\mr\8\）

PHP 所支持的数据库类型较多，在这些数据库中，MySQL 数据库与 PHP 结合最好，与 Linux 系统、Apache 服务器和 PHP 语言构成了当今主流的 LAMP 网站架构模式，并且 PHP 提供了多种操作 MySQL 数据库的方式，从而适合不同需求和不同类型项目的需要。本章将系统讲解如何通过 PHP 内置的 MySQL 函数库操作 MySQL 数据库。

学习摘要：

- PHP 操作 MySQL 数据库的步骤
- PHP 操作 MySQL 数据库的函数
- 向 MySQL 数据库中添加数据
- 浏览 MySQL 数据库中的数据
- 编辑 MySQL 数据库中的数据
- 删除 MySQL 数据库中的数据
- 查询 MySQL 数据库中的数据





8.1　PHP 操作 MySQL 数据库的步骤

MySQL 是一款广受欢迎的数据库，由于它是开源的半商业软件，所以市场占有率高，备受 PHP 开发者的青睐，一直被认为是 PHP 最好的搭档。PHP 具有强大的数据库支持能力，本节主要讲解 PHP 操作 MySQL 数据库的步骤。

PHP 操作 MySQL 数据库的步骤如图 8.1 所示。

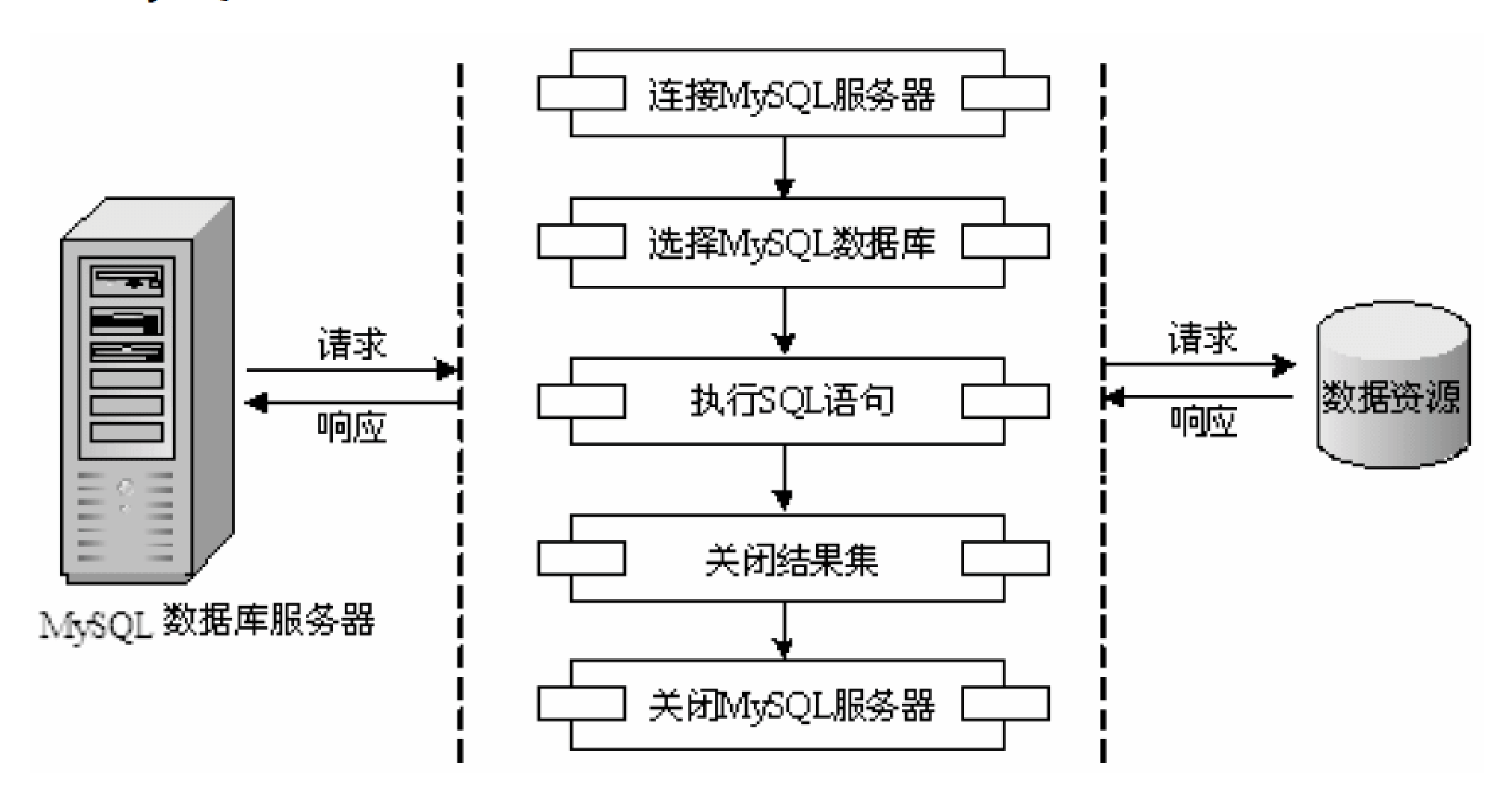


图 8.1　PHP 操作 MySQL 数据库的步骤

8.2　PHP 操作 MySQL 数据库的函数

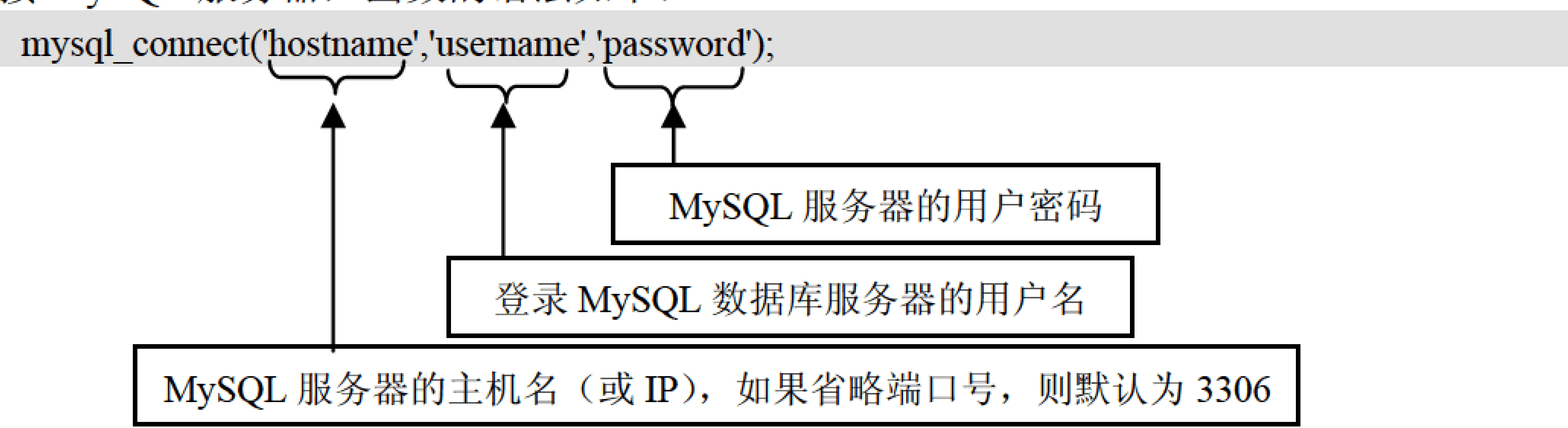
视频讲解：配套资源\mr\8\video\PHP 操作 MySQL 数据库的函数.exe

PHP 中提供了很多操作 MySQL 数据库的函数，使用这些函数可以对 MySQL 数据库执行各种操作，使程序开发变得更加简单、灵活。下面介绍一些常用的 MySQL 数据库函数。

8.2.1　mysql_connect()函数连接 MySQL 服务器

要操作 MySQL 数据库，必须先与 MySQL 服务器建立连接。PHP 中通过 mysql_connect()函数连接 MySQL 服务器，函数的语法如下：

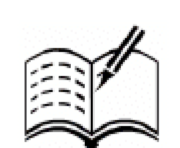
```
mysql_connect('hostname','username','password');
```



该函数的返回值用于表示这个数据库连接。如果连接成功，则函数返回一个连接标识，失败则返回 False。例如，使用 mysql_connect()函数连接本地 MySQL 服务器，代码如下：



```
<?php
$conn = mysql_connect("localhost", "root", "111") or die("连接数据库服务器失败！".mysql_error());
?>
```



Note

为了方便查询因为连接问题而出现的错误，采用 die()函数生成错误处理机制，使用 mysql_error()函数提取 MySQL 函数的错误文本，如果没有出错，则返回空字符串。如果浏览器显示“Warning: mysql_connect()……”的字样，则说明是数据库连接的错误，这样就能迅速地发现错误位置，并及时改正。

多学两招：

在 mysql_connect()函数前面添加符号“@”，用于限制这个命令的出错信息的显示。如果函数调用出错，将执行 or 后面的语句。die()函数表示向用户输出引号中的内容后，程序终止执行。这样是为了防止数据库连接出错时，用户看到一堆莫名其妙的专业名词，而是提示定制的出错信息。但在调试时不要屏蔽出错信息，避免出错后难以找到问题。

8.2.2　mysql_select_db()函数选择 MySQL 数据库

与 MySQL 服务器建立连接后要确定所要连接的数据库，使用 mysql_select_db()函数可以连接 MySQL 服务器中的数据库，函数语法如下：

```
mysql_select_db ( string数据库名[,resource link_identifier] )
```

string数据库名：选择的 MySQL 数据库名称

[,resource link_identifier]：MySQL 服务器的连接标识

例如，与本地 MySQL 服务器中的 db_database08 数据库建立连接，代码如下：

```
<?php
$conn=mysql_connect("localhost","root","111");          //连接mysql数据库服务器
$select=mysql_select_db("db_database08",$conn);         //连接服务器中的db_database08表
if($select){                                            //判断是否连接成功
    echo "数据库连接成功！";
}
?>
```

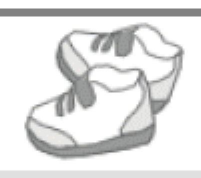
如果数据库连接成功，则输出“数据库连接成功！”。

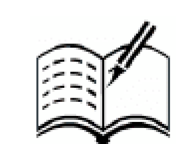
多学两招：

在开发一个完整的 Web 程序过程中，经常需要连接数据库，如果总是重复编写代码，会造成代码的冗余，而且不利于程序维护，所以通常将连接 MySQL 数据库的代码单独建立一个名为 conn.php 的文件，存储在根目录下的 conn 文件夹中，通过 require 语句包含这个文件即可。

8.2.3　mysql_query()函数执行 SQL 语句

在 PHP 中，通常使用 mysql_query()函数来执行对数据库操作的 SQL 语句。mysql_query()函数的语法如下：






```
mysql_query ( string query [, resource link_identifier] )
```

参数 query 是传入的 SQL 语句，包括插入数据（insert）、修改记录（update）、删除记录（delete）、查询记录（select）；参数 link_identifier 是 MySQL 服务器的连接标识。

例如，向会员信息表 tb_user 中插入一条会员记录，SQL 语句的代码如下：

```
$result=mysql_query("insert into tb_user values('mr','111')",$conn);
```

例如，修改会员信息 tb_user 表中的会员记录，SQL 语句的代码如下：

```
$result=mysql_query("update tb_user set name='lx' where id='01'",$conn);
```

例如，删除会员信息 tb_user 表中的一条会员记录，SQL 语句的代码如下：

```
$result=mysql_query("delete from tb_user where name='mr'",$conn);
```

例如，查询会员信息 tb_user 表中 name 字段值为 mr 的记录，SQL 语句的代码如下：

```
$result=mysql_query("select * from tb_user where name='mr'",$conn);
```

上述 SQL 语句代码都是将结果赋给变量$result。

8.2.4　mysql_fetch_array()函数将结果集返回到数组中

使用 mysql_query()函数执行 select 语句时，成功将返回查询结果集，返回结果集后，使用 mysql_fetch_array()函数可以获取查询结果集信息，并放入到一个数组中。函数语法如下：

```
array mysql_fetch_array ( resource result [, int result_type] )
```

参数 result：资源类型的参数，要传入的是由 mysql_query()函数返回的数据指针。

参数 result_type：可选参数，设置结果集数组的表述方式，默认值是 MYSQL_BOTH。其可选值如下：

☑ MYSQL_ASSOC：表示数组采用关联索引。

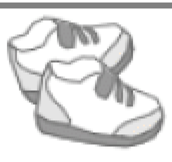
☑ MYSQL_NUM：表示数组采用数字索引。

☑ MYSQL_BOTH：同时包含关联和数字索引的数组。

例 8.1　获取新闻信息表 tb_news 中的新闻信息，并使用 mysql_fetch_array()函数返回结果集，然后使用 while 语句循环输出新闻标题及内容，代码如下：（**实例位置：配套资源\mr\8\example\8.1**）

```
<?php
/*连接数据库*/
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                 //选择数据库
mysql_query("set names utf8");                          //设置编码格式
$arr=mysql_query("select * from tb_news",$conn);
/*使用while语句循环mysql_fetch_array()函数返回的数组*/
while($result=mysql_fetch_array($arr)){
?>
        <tr>
          <td height="25"><?php echo $result['name'];?><!--输出新闻标题--> </td>
          <td height="25"><?php echo $result['news'];?><!--输出新闻内容--> </td>
          </tr>
<?php
  }             //结束while循环
?>
```

运行结果如图 8.2 所示。



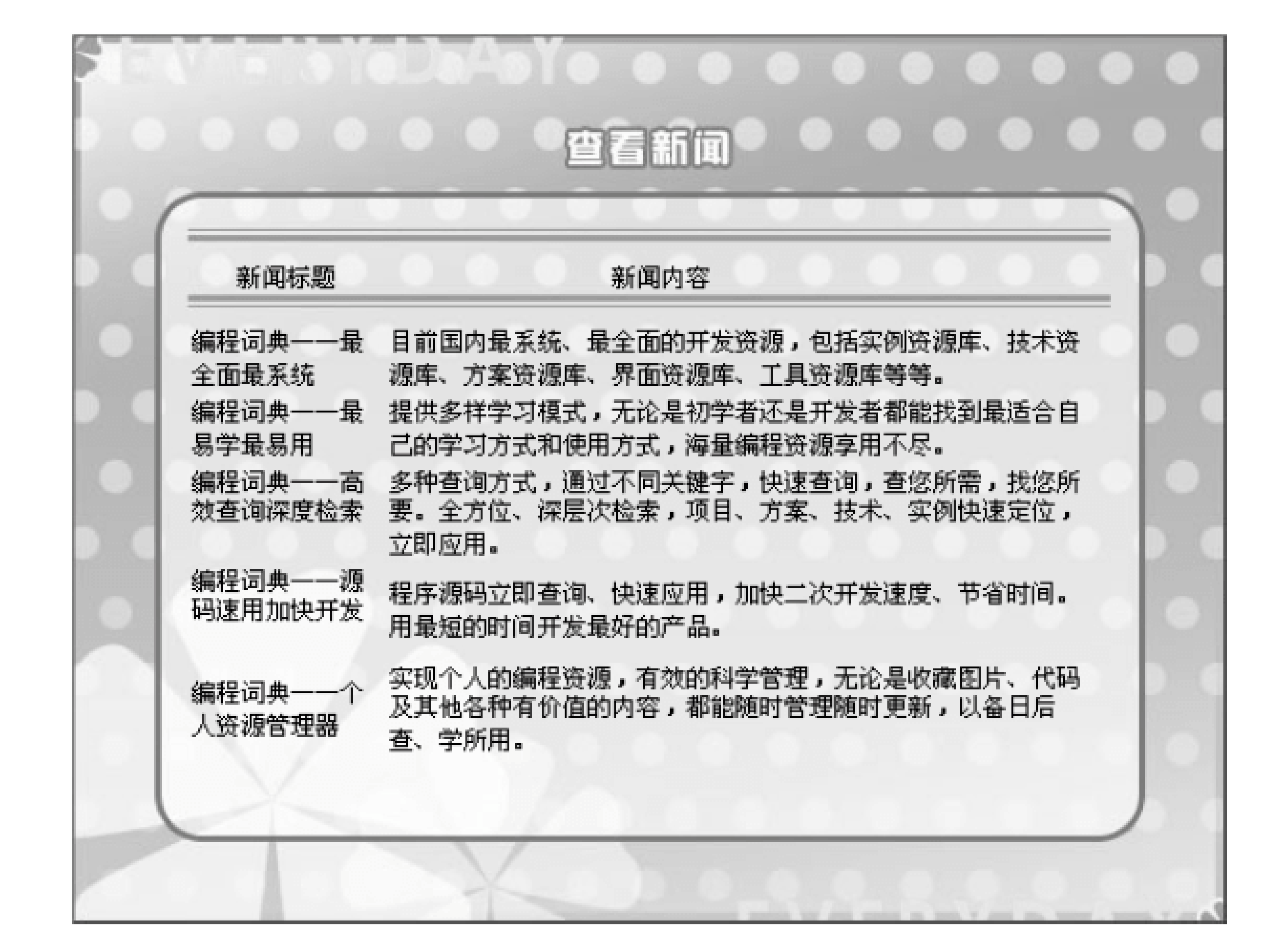



图 8.2　查看新闻内容

8.2.5　mysql_fetch_row()函数从结果集中获取一行作为枚举数组

mysql_fetch_row()函数从结果集中取得一行作为枚举数组。在应用 mysql_fetch_row()函数逐行获取结果集中的记录时，只能使用数字索引来读取数组中的数据，其语法如下：

```
array mysql_fetch_row ( resource result )
```

mysql_fetch_row()函数返回根据所取得的行生成的数组，如果没有更多行则返回 False。返回数组的偏移量从 0 开始，即以$row[0]的形式访问第一个元素（只有一个元素时也是如此）。

例 8.2　获取 db_database08 数据库的 tb_news 数据表中的新闻信息，但是与 8.2.4 节不同的是，本例应用 mysql_fetch_row()函数逐行获取结果集中的记录，并通过 while 语句循环输出查询结果集，其代码如下：**（实例位置：配套资源\mr\8\example\8.2）**

```
<?php
/*连接数据库*/
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                 //选择数据库
mysql_query("set names utf8");                          //设置编码格式
$arr=mysql_query("select * from tb_news",$conn);
/*使用while语句循环mysql_fetch_row()函数返回的数组*/
while($result=mysql_fetch_row($arr)){
?>
          <tr>
            <td height="25"><?php echo $result[1];?>  </td>    <!--输出新闻标题-->
            <td height="25"><?php echo $result[2];?> </td>          <!--输出新闻内容-->
          </tr>
<?php
  }                                                     //结束while循环
?>
```

运行结果如图 8.3 所示。





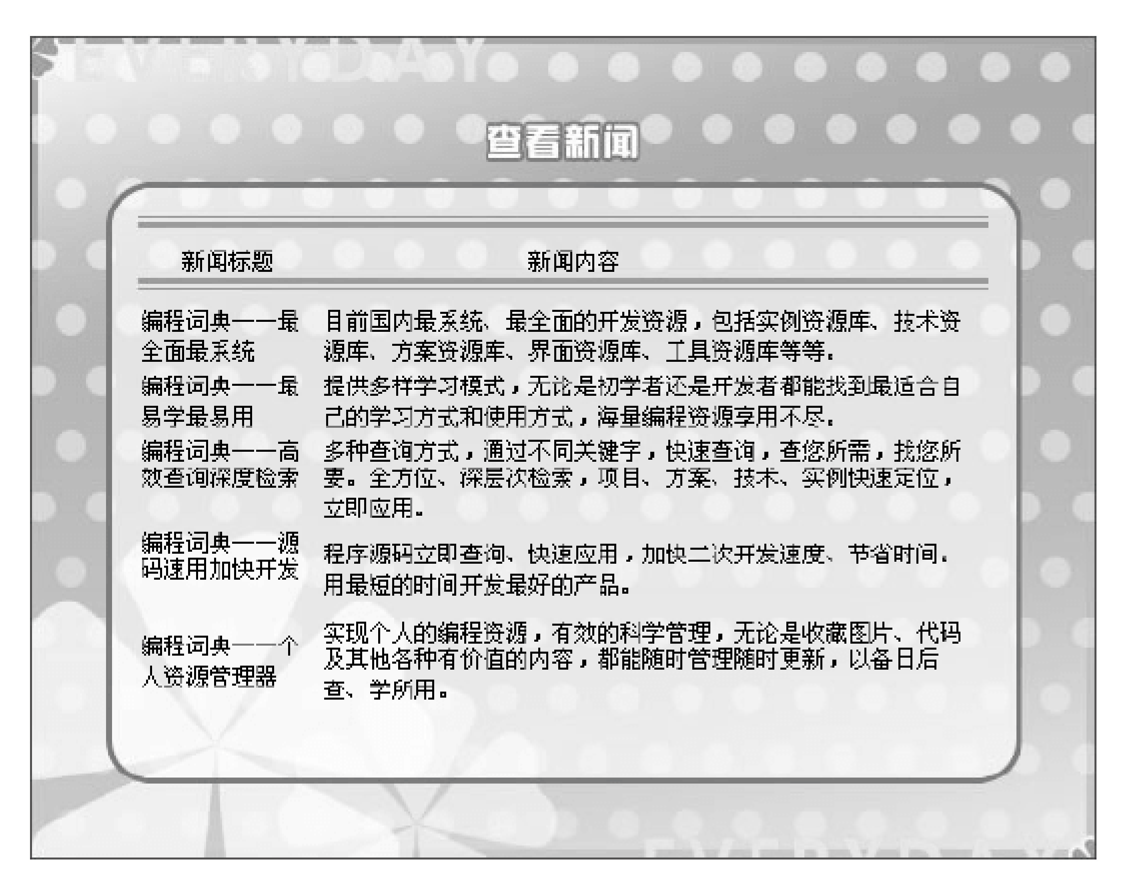


图 8.3　查看新闻内容

指点迷津：

mysql_fetch_array()函数和 mysql_fetch_row()函数的区别在于：

使用 mysql_fetch_array()函数获取到的数组可以是数字索引数组，也可以是关联数组，而使用 mysql_fetch_row()函数获取到的数组，只能是数字索引数组。

8.2.6　mysql_num_rows()函数获取查询结果集中的记录数

使用 mysql_num_rows()函数可以获取由 select 语句查询到的结果集中行的数目，该函数的语法如下：

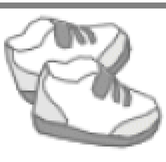
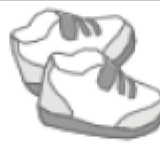
```
int mysql_num_rows ( resource result )
```

此命令仅对 SELECT 语句有效。要取得被 INSERT、UPDATE 或 DELETE 语句所影响到的行的数目，则需使用 mysql_affected_rows()函数。

例 8.3　使用 mysql_num_rows()函数获取查询到的新闻条数，关键代码如下：（**实例位置：配套资源\mr\8\example\8.3**）

```
<?php
/*连接数据库*/
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                 //选择数据库
mysql_query("set names utf8");                          //设置编码格式
$arr=mysql_query("select * from tb_news",$conn);
/*使用while语句循环mysql_fetch_array()函数返回的数组*/
while($result=mysql_fetch_array($arr)){
?>
        <tr>
          <td height="25"><?php echo $result['name'];?><!--输出新闻标题--> </td>
          <td height="25"><?php echo $result['news'];?><!--输出新闻内容--></td>
        </tr>
<?php
  }            //结束while循环
?>
<?php echo mysql_num_rows($arr);?> 条          //查询新闻总数
```

运行结果如图 8.4 所示。

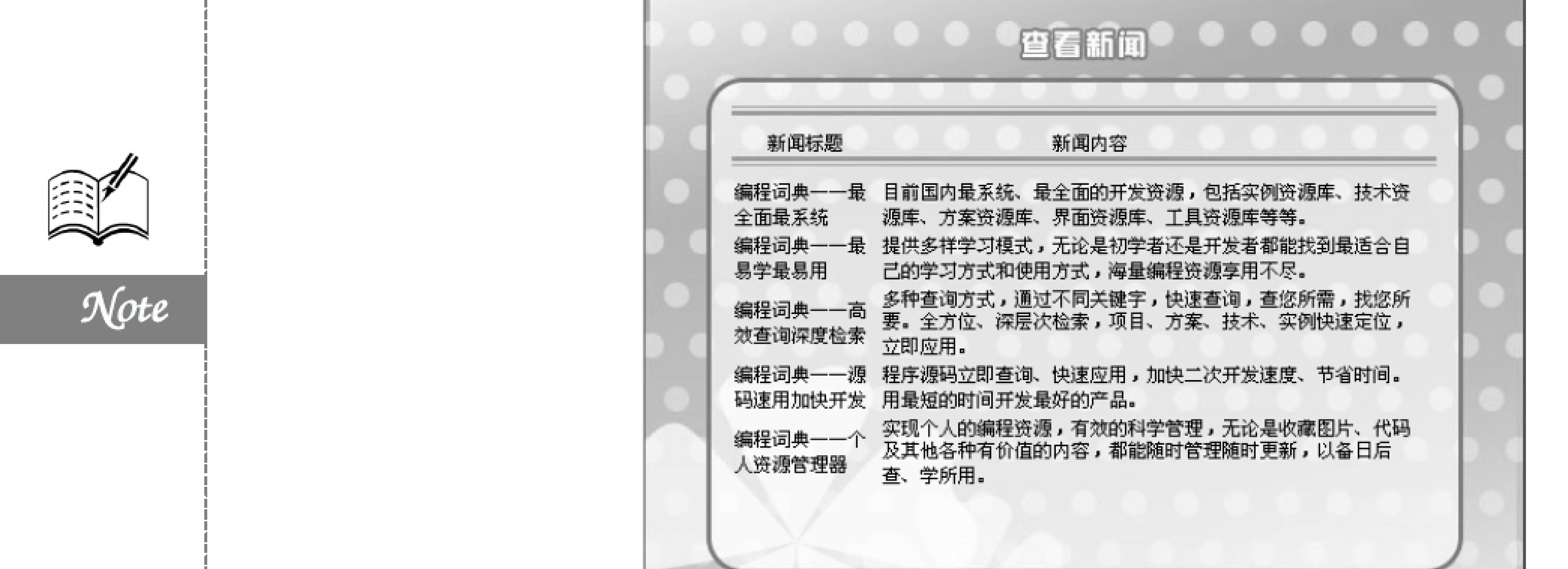


图 8.4　获取查询到的新闻数

8.2.7　mysql_free_result()函数释放内存

mysql_free_result()函数用于释放内存，数据库操作完成后需要关闭结果集，以释放系统资源。该函数的语法如下：

```
mysql_free_result($result);
```

mysql_free_result()函数将释放所有与结果标识符 result 所关联的内存。该函数仅需要在考虑到返回很大的结果集且占用多少内存时调用。在脚本结束后所有关联的内存都会被自动释放。

下面举一个使用该函数释放内存的例子，具体代码如下：

```
<?php
mysql_connect("localhost", "root", "111");                          //连接数据库服务器
mysql_select_db("db_mrbook");                                       //选择数据库
$result = mysql_query("select * from tb_book");                     //执行查询语句
while($row=mysql_fetch_array($result)) {                            //循环输出查询结果
    printf ("ID: %s    BookName: %s",$row[0],$row["bookname"])."<br>"; //打印输出具体内容
}
mysql_free_result($result);                                         //释放结果内存空间
?>
```

本示例的运行结果如下：

```
ID: 2 BookName: PHP 程序开发范例宝典
ID: 3 BookName: PHP+MySQL 数据库系统开发完全手册
ID: 4 BookName: PHP+MySQL 数据库系统开发完全手册
ID: 11 BookName: PHP 程序开发范例宝典
```

多学两招：

如果在多个网页中都要频繁进行数据库访问，则可以建立与数据库服务器的持续连接来提高效率，因为每次与数据库服务器的连接需要较长的时间和较大的资源开销，持续的连接相对来说会更有效。建立持续连接的方法就是在数据库连接时，调用函数 mysql_pconnect()代替 mysql_connect()函数。建立的持续连接在本程序结束时，不需要调用 mysql_close()来关闭。当下次程序再次执行 mysql_pconnect()函数时，系统将自动直接返回已经建立的持续连接的 ID 号，而不再去真的连接数据库。



Note

8.2.8　mysql_close()函数关闭连接

每使用一次 mysql_connect()或 mysql_query()函数，都会消耗系统资源。在少量用户访问 Web 网站时问题还不大，但如果用户连接超过一定数量时，就会造成系统性能的下降，甚至死机。为了避免这种现象的发生，在完成数据库的操作后，应使用 mysql_close()函数关闭与 MySQL 服务器的连接，以节省系统资源。mysql_close()函数的语法如下：

```
mysql_close($conn);
```

在 Web 网站的实际项目开发过程中，经常需要在 Web 页面中查询数据信息，查询后使用 mysql_close()函数关闭数据源。

例 8.4　在浏览新闻信息页面中，浏览信息后应用 mysql_close()函数关闭已经连接的数据源，代码如下：**（实例位置：配套资源\mr\8\example\8.4）**

```
<?php
/*连接数据库*/
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                  //选择数据库
mysql_query("set names utf8");                           //设置编码格式
$arr=mysql_query("select * from tb_news",$conn);        //执行查询语句
/*使用while语句循环mysql_fetch_array()函数返回的数组*/
while($result=mysql_fetch_array($arr)){                  //循环输出查询结果
?>
          <tr>
            <td height="25"><?php echo $result['name'];?><!--输出新闻标题--></td>
            <td height="25"><?php echo $result['news'];?><!--输出新闻内容--></td>
          </tr>
<?php
}                                                        //结束while循环
mysql_close($conn);                                      //关闭数据源
?>
```

多学两招：

PHP 中与数据库的连接是非持久连接，系统会自动回收，一般不用设置关闭。但如果一次性返回的结果集比较大或网站访问量比较多，则最好使用 mysql_close()函数手动进行释放。

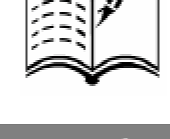
上机演练

上机演练 1　通过 MySQL 函数访问数据库

本例中通过 PHP 提供的 MySQL 函数来访问 MySQL 数据库。这里以开发一个电子商务网站为背景，通过 MySQL 函数实现与 MySQL 数据库的连接，然后读取 MySQL 数据库中的图书商品信息，并且将图书产品信息输出到页面中，运行结果如图 8.5 所示。

上机演练 2　将数据以二进制形式上传到数据库

在进行网站开发的过程中，有一个问题是必须要考虑的，就是网站是否支持文件上传的功能？如果支持，那么又该将文件存储于什么地方？是存储在服务器中还是存储在数据库中？具



Note

体将数据存储于什么地方，应该根据实际情况具体问题具体分析，需要考虑文件的大小、类型等因素。在电子商务网站中，每种商品在展示的过程中都应该有效果图，而这个效果图的大小基本不会超过 5MB，类型为 bmp、jpg 或 gif。对于这样的文件，我们就可以将其存储到数据库中。本例中将电子商务网站的商品效果图以二进制的形式存储到 MySQL 数据库中，并且输出上传的图书，运行结果如图 8.6 所示。



图 8.5　通过 MySQL 函数访问数据库



图 8.6　将数据以二进制形式上传到数据库

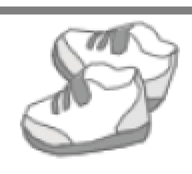
8.3　管理 MySQL 数据库中的数据

视频讲解：配套资源\mr\8\video\管理 MySQL 数据库中的数据.exe

管理 MySQL 数据库中的数据主要是对数据进行添加、修改、删除、查询等操作，只有熟练地掌握这部分知识才能够独立开发出基于 PHP 的数据库项目。

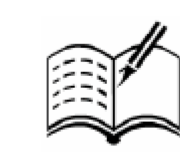
8.3.1　向数据库中添加数据

向数据库中添加数据主要通过 mysql_query()函数和 insert 语句实现。



例 8.5　发表博客文章，填写文章标题及文章内容，当用户单击“提交”按钮时，判断文章标题及内容是否为空，如果不为空则将数据添加到数据库中，关键代码如下：（**实例位置：配套资源\mr\8\example\8.5**）

```
<?php
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                 //选择数据库
mysql_query("set names utf8");                          //设置编码格式
if(isset($_POST['Submit']) and $_POST['textfield']!=null and $_POST['Submit']=="提交"){
    $insert=mysql_query("insert into tb_news(name,news) values('".$_POST['textfield']."','".$_POST
['textarea']."')");
    if($insert){
        echo "<script> alert('发表成功!'); window.location.href='index.php'</script>";
    }else{
        echo "<script> alert('发表失败!'); window.location.href='index.php'</script>";
    }
}else{
    echo "<script> alert('发表失败!'); window.location.href='index.php'</script>";
}
?>
```



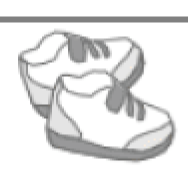
运行结果如图 8.7 所示。



图 8.7　添加文章

8.3.2　浏览数据库中数据

浏览数据库中的数据通过 mysql_query()函数和 select 语句查询数据，并使用 mysql_fetch_assoc()函数将查询结果返回到数组中。



例 8.6 浏览表 tb_news 中的新闻信息，具体代码如下：（**实例位置：配套资源\mr\8\example\8.6**）

```
<?php
/*连接数据库*/
$conn=mysql_connect("localhost","root","111");            //连接数据库服务器
mysql_select_db("db_database08",$conn);                    //选择数据库
mysql_query("set names utf8");                             //设置编码格式
$arr=mysql_query("select * from tb_news",$conn);           //执行查询语句
/*使用while语句循环mysql_fetch_assoc()函数返回的数组*/
while($result=mysql_fetch_assoc($arr)){                    //循环输出查询结果
?>
    <tr>
        <td height="25"><?php echo $result['name'];?>  </td>         <!--输出新闻标题-->
         <td height="25"><?php echo $result['news'];?> </td>              <!--输出新闻内容-->
    </tr>
<?php
  }                                                        //结束while循环
?>
```

运行结果如图 8.8 所示。

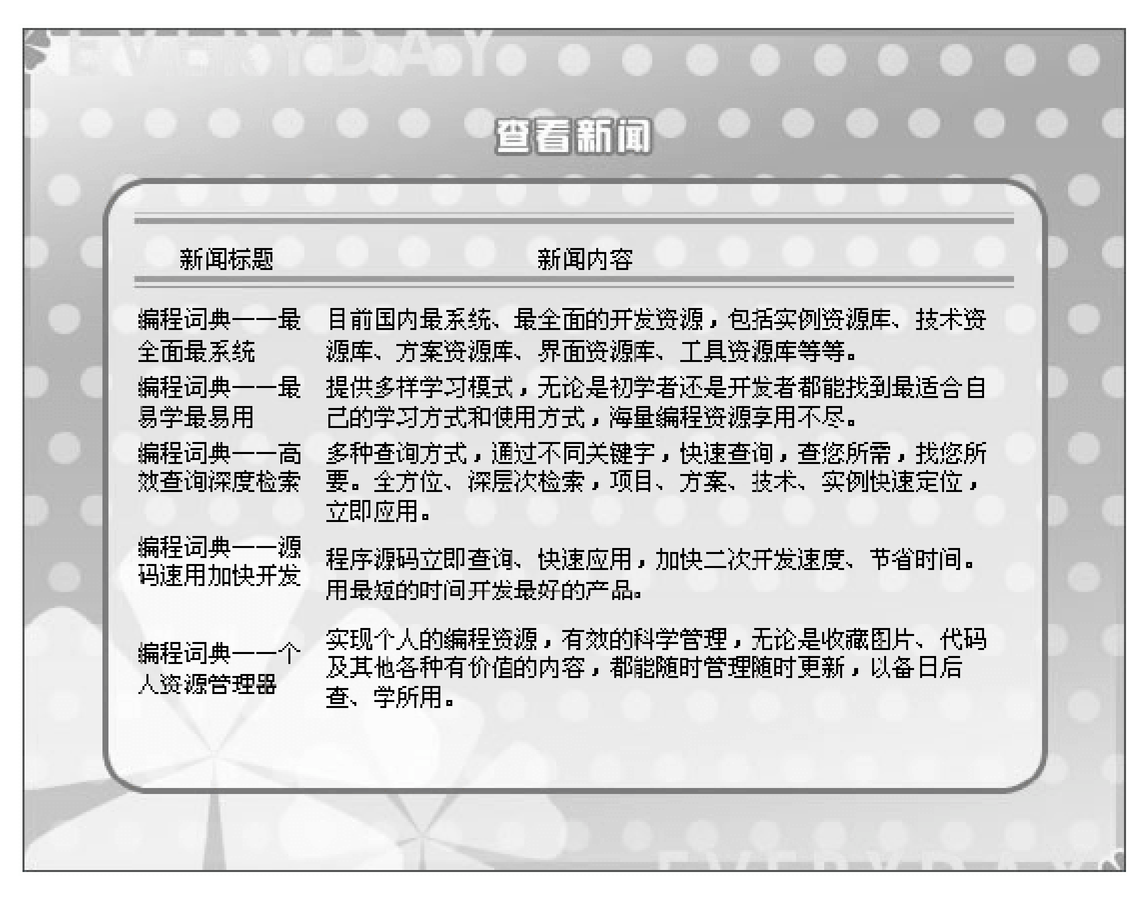


图 8.8 浏览新闻信息

8.3.3 编辑数据库数据

编辑数据库数据主要通过 mysql_query()函数和 update 语句实现。

例 8.7 编辑新闻信息表中的新闻信息，具体步骤如下：（**实例位置：配套资源\mr\8\example\8.7**）

（1）创建数据库连接文件 conn.php，代码如下：

```
<?php
$conn=mysql_connect("localhost","root","111");            //连接数据库服务器
mysql_select_db("db_database08",$conn);                    //连接db_database08数据库
mysql_query("set names utf8");                             //设置数据库编码格式
?>
```





（2）创建 index.php 文件，显示所有新闻信息，代码如下：

```
<?php
include("conn.php");                                    //包含conn.php文件
$arr=mysql_query("select * from tb_news",$conn);        //查询数据
/*使用while语句循环mysql_fetch_array()函数返回的数组*/
while($result=mysql_fetch_array($arr)){
?>
        <tr>
          <td height="25"><?php echo $result['name'];?>     <!--输出新闻标题--> </td>
          <td><?php echo $result['news'];?>                 <!--输出新闻内容-->        </td>
          <td><label>
            <input type="hidden" name="id" value="<?php echo $result['id'];?>" />
            <div align="center"><a href="update.php?id=<?php echo $result['id'];?>">编辑</a></div>
          </label></td>
          </tr>
<?php
  }                                                     //结束while循环
?>
```

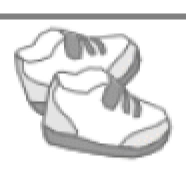
（3）创建 update.php 文件，显示要编辑的新闻内容，代码如下：

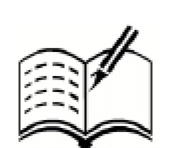
```
<form id="form1" name="form1" method="post" action="update_ok.php">
<?php
include("conn.php");                    //包含conn.php文件
$arr=mysql_query("select * from tb_news where id='".$_GET['id']."'",$conn);     //定义查询语句
$select=mysql_fetch_array($arr);                        //循环输出查询内容
?>
    <input name="name" type="text" size="40" value="<?php echo $select['name'];?>"/>
    <textarea name="news" cols="40" rows="10"><?php echo $select['news'];?></textarea>
    <input type="submit" name="Submit" value="保存" />
    <input type="hidden" name="id" value="<?php echo $select['id'];?>" />
</form>
```

（4）创建 update_ok.php 文件，完成新闻信息的编辑操作，代码如下：

```
<?php
include("conn.php");                                    //包含conn.php文件
if(isset($_POST['id']) and isset($_POST['Submit']) and $_POST['Submit']=="保存"){
    $update=mysql_query("update tb_news set name='".$_POST['name']."',news='".$_POST['news']."'
where id='".$_POST['id']."'",$conn);
    if($update){
        echo   "<script> alert('修改成功!'); window.location.href='index.php'</script>";
    }else{
        echo   "<script> alert('修改失败!'); window.location.href='index.php'</script>";
    }
}
?>
```

运行结果如图 8.9 所示。





Note

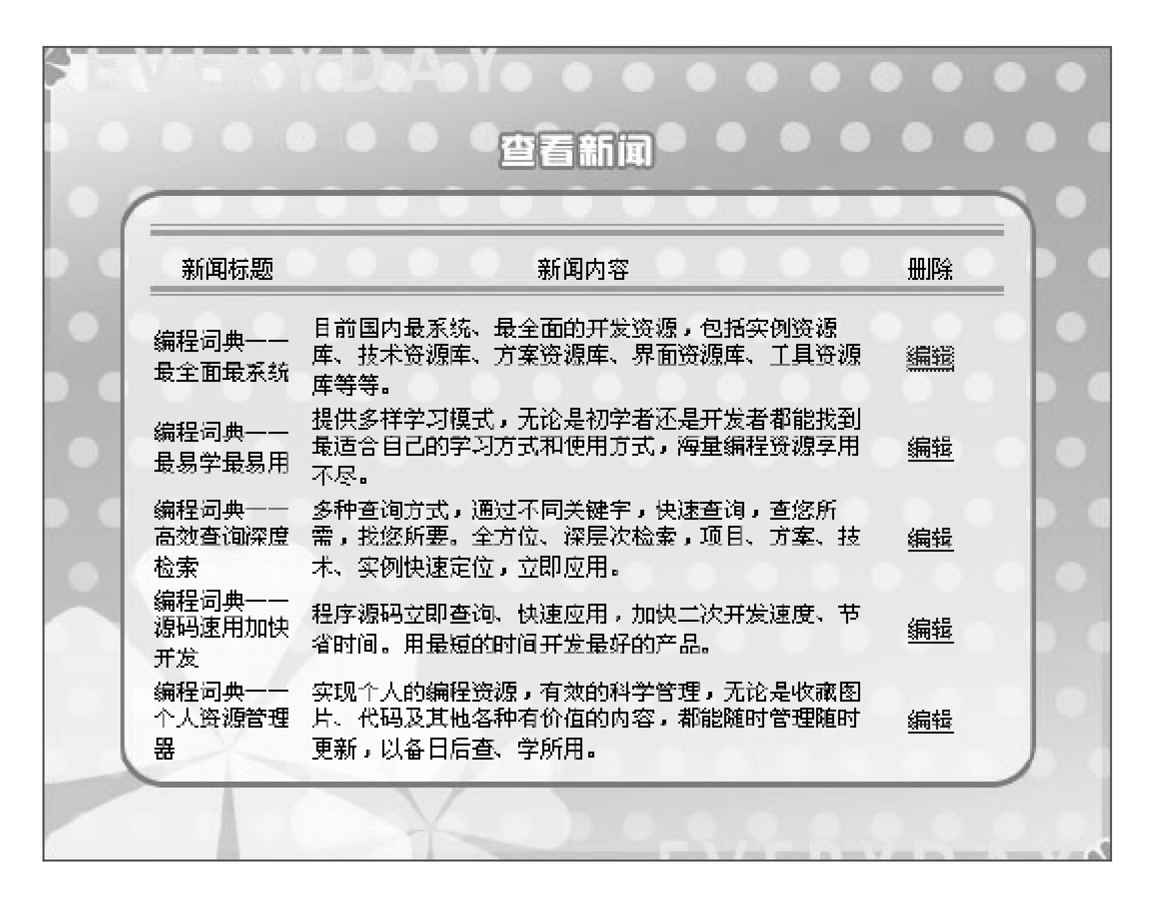


图 8.9　编辑新闻信息

8.3.4　删除数据

数据的删除应用 delete 语句，而在 PHP 中需要通过 mysql_query()函数来执行这个 delete 删除语句，完成 MySQL 数据库中数据的删除操作。

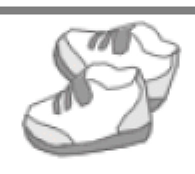
例 8.8　删除新闻信息表中的新闻信息，具体步骤如下。（**实例位置：配套资源\mr\8\example\8.8**）

（1）创建数据库连接文件 conn.php，代码如下：

```
<?php
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                  //连接db_database08数据库
mysql_query("set names utf8");                           //设置数据库编码格式
?>
```

（2）创建 index.php 文件，显示所有新闻信息，代码如下：

```
<?php
include("conn.php");                                          //包含conn.php文件
$arr=mysql_query("select * from tb_news",$conn);              //查询数据
/*使用while语句循环mysql_fetch_array()函数返回的数组*/
while($result=mysql_fetch_array($arr)){
?>
        <tr>
          <td height="25"><?php echo $result['name'];?>     <!--输出新闻标题--> </td>
          <td><?php echo $result['news'];?>                 <!--输出新闻内容-->       </td>
          <td><label>
            <input type="hidden" name="id" value="<?php echo $result['id'];?>" />
            <div align="center">
              <input name="Submit" type="submit" class="STYLE3" value="删除" />
              </div>
          </label></td>
          </tr>
<?php
  }                                                           //结束while循环
?>
```





（3）创建 index_ok.php 文件，根据超链接传递的 ID 值完成新闻信息的删除操作，代码如下：

```
<?php
include("conn.php");                                          //包含conn.php文件
if(isset($_POST['id']) and isset($_POST['Submit']) and $_POST['Submit']=="删除"){
    $delete=mysql_query("delete from tb_news where id='".$_POST['id']."'",$conn);//执行删除操作
    if($delete){
        echo   "<script> alert('删除成功!'); window.location.href='index.php'</script>";
    }else{
        echo   "<script> alert('删除失败!'); window.location.href='index.php'</script>";
    }
}
?>
```

执行程序，运行结果如图 8.10 所示。

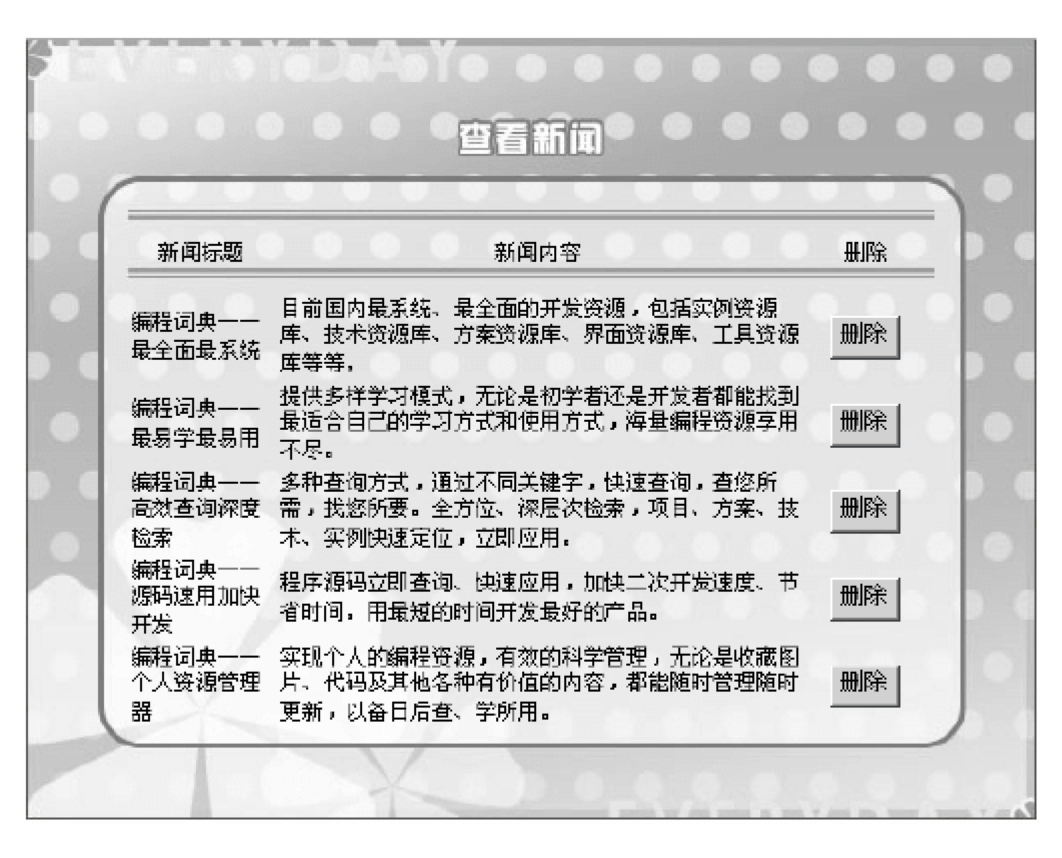


图 8.10　删除新闻信息

8.3.5　批量删除数据

在对数据库中的数据进行管理的过程中，如果要删除的数据非常多，则执行单条删除数据的操作就显得很不适合，这时应该使用批量删除数据的方法来实现数据库中信息的删除。通过数据的批量删除可以快速地删除多条数据，从而减少操作执行的时间。

例 8.9　批量清理新闻信息表中陈旧的新闻信息，具体步骤如下。（**实例位置：配套资源\mr\8\example\8.9**）

（1）创建数据库连接文件 conn.php，代码如下：

```
<?php
$conn=mysql_connect("localhost","root","111");          //连接数据库服务器
mysql_select_db("db_database08",$conn);                 //连接db_database08数据库
mysql_query("set names utf8");                          //设置数据库编码格式
?>
```

（2）创建 index.php 文件，显示所有的新闻信息，代码如下：

```
<?php
include("conn.php");
$arr=mysql_query("select * from tb_news",$conn);        //查询数据
/*使用while语句循环mysql_fetch_array()函数返回的数组*/
while($result=mysql_fetch_array($arr)){
```



```
?>
            <tr>
                <td><label>
                    <label>
                    <input type="checkbox" name="checkbox[]" value="<?php echo $result['id'];?>" />
                    </label>
                </label></td>
                <td height="25"><?php echo $result['name'];?><!--输出新闻标题--></td>
                <td><?php echo $result['news'];?><!--输出新闻内容--></td>
            </tr>
<?php
    }              //结束while循环
?>
```

（3）创建 index_ok.php 页面，完成批量删除操作，代码如下：

```
<?php
include("conn.php");                          //包含conn.php文件
if(isset($_POST['Submit']) and $_POST['Submit']=="删除" and $_POST['checkbox']!=""){    //判断是否执行删除操作
    for($i=0;$i<count($_POST['checkbox']);$i++){                //遍历复选框获取到的新闻id序号
        $sql=mysql_query("delete from tb_news where id='".$_POST['checkbox'][$i]."'",$conn);//执行删除操作
    }
    if($sql){
        echo  "<script> alert('删除成功!'); window.location.href='index.php'</script>";
    }else{
        echo  "<script> alert('删除失败!'); window.location.href='index.php'</script>";
    }
}else{
    echo  "<script> alert('请选择要删除的内容!'); window.location.href='index.php'</script>";
}
?>
```

执行程序，运行结果如图 8.11 所示。

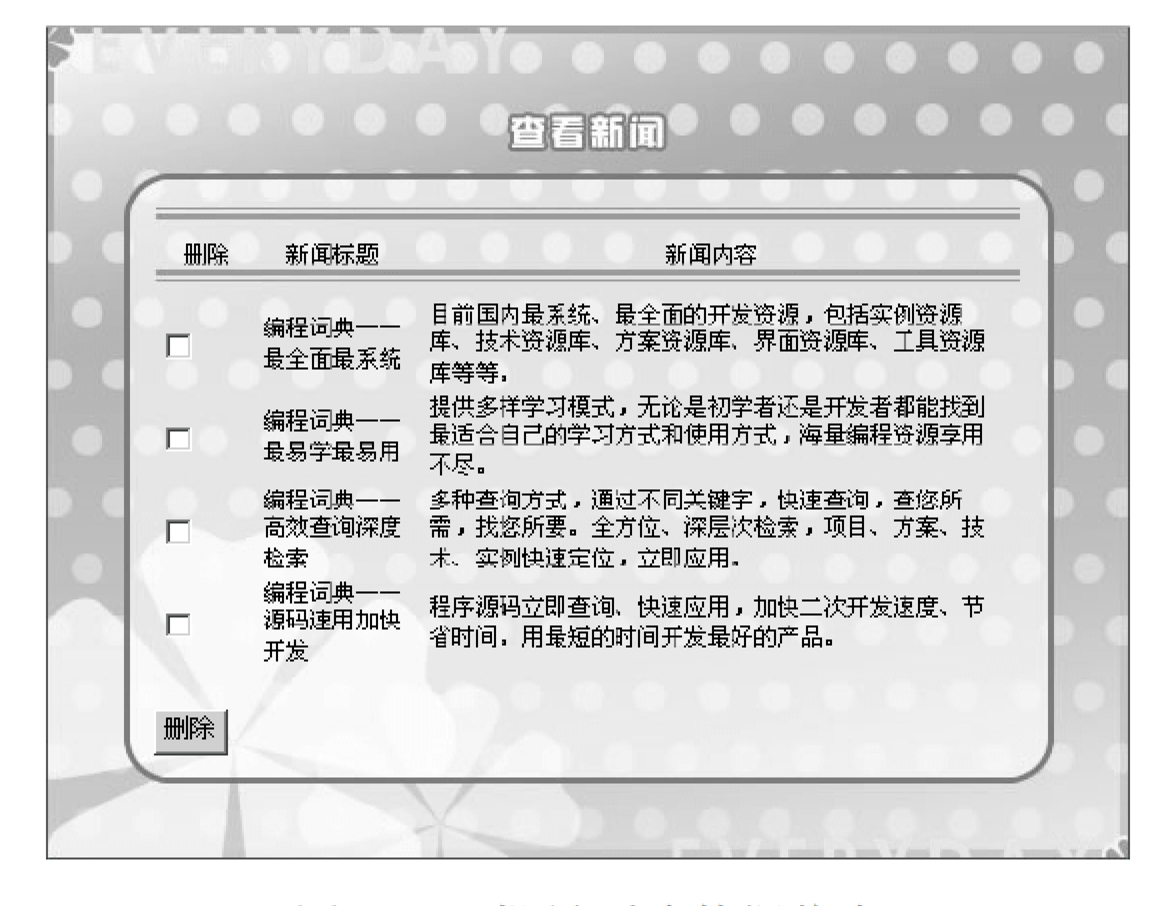
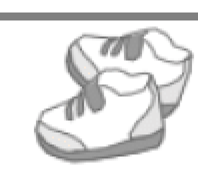


图 8.11　批量删除数据信息





上机演练

上机演练 3　查询日期型数据

在商业网站中对日期型数据进行查询得到了广泛应用，例如查询某范围员工的出生日期、商品的进货时间等。本例中实现在图书信息表中查询图书出版日期为“2010-07-19”的图书信息，其运行结果如图 8.12 所示。

上机演练 4　查询字符串

对字符串进行查询是项目开发过程中应用几率最高的查询，并且这种查询经常与通配符配合使用实现信息的匹配查询。本例中根据提交的图书名称进行查询，并且输出查询结果，其运行结果如图 8.13 所示。



查询日期型数据

查询数据

ID	书名	剩余	日期
1	《PHP范例大全》	101	2010-07-19
2	《JAVA范例大全》	150	2010-07-19
3	《VB范例大全》	180	2010-07-19
4	《C++范例大全》	178	2010-07-19

图 8.12　查询日期型数据

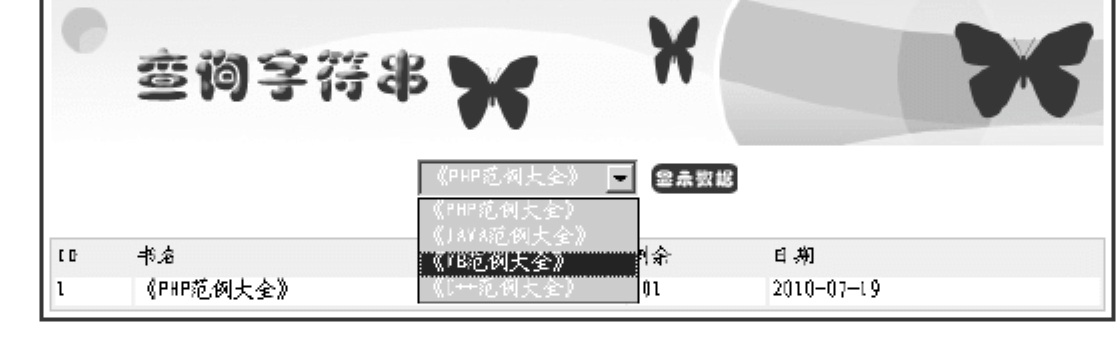


图 8.13　查询字符串

上机演练 5　使用 MySQL 存储过程实现用户注册

存储过程是 MySQL5.0 以后的版本中才支持的技术，通过存储过程的应用可以提高系统的运行效率。本例中应用存储过程实现用户注册功能，其运行效果如图 8.14 所示。在该图的文本框中输入注册信息后，单击“注册”按钮即可将用户的注册信息保存到数据库中。

上机演练 6　使用 MySQL 事务处理实现银行安全转账

事务处理机制在程序开发过程中有着非常重要的作用，它可以使整个系统更加安全，例如在银行处理转账业务时，如果 A 账户中的金额刚被发出，而 B 账户还没来得及接受就发生停电，这会给银行和个人带来很大的经济损失。采用事务处理机制，一旦在转账过程中发生意外，则程序将回滚，不做任何处理。

本例中设计了一个模拟的银行转账系统，并且应用事务处理机制保证转账操作安全、顺利的进行，其运行效果如图 8.15 所示。在该图的文本框中输入要转给 B 账户的金额后，单击“转账”按钮即可实现转账。



图 8.14　填写用户注册信息

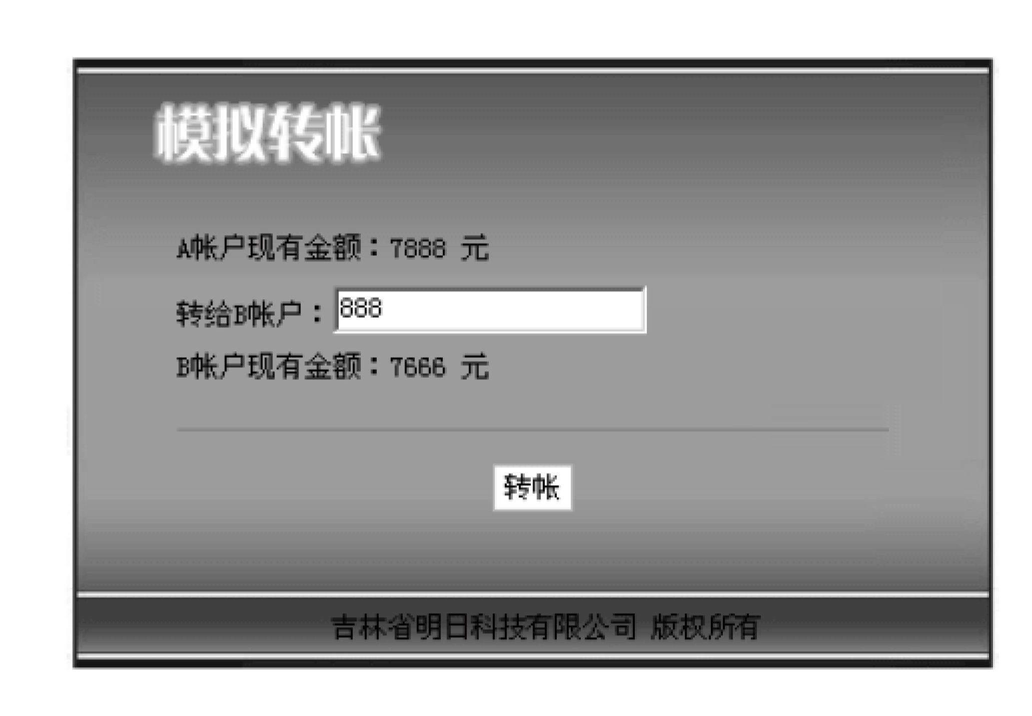
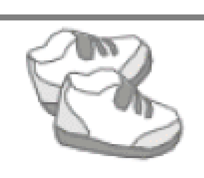


图 8.15　应用事务回滚模拟银行转账





本 章 摘 要

1．mysql_connect()函数连接 MySQL 服务器。

2．mysql_select_db()函数选择 MySQL 数据库。

3．mysql_query()函数执行 SQL 语句。

4．mysql_fetch_array()函数获取查询结果集，mysql_fetch_row()函数获取结果集中的一行数据，mysql_num_rows()函数统计结果集中的记录数。

5．mysql_free_result()函数释放内存。

6．mysql_close()函数关闭连接。

7．详细讲解通过 MySQL 函数操作 MySQL 数据库的方法，包括数据的添加、浏览、编辑和删除。

习 题

1．连接 MySQL 数据库服务器的函数是（　　）。

A．mysql_connect()　　B．mysql_query()

C．mysql_error()　　D．mysql_select_db()

2．获取查询结果记录数使用函数（　　）。

A．mysql_fetch_array()　　B．mysql_fetch_rows()

C．mysql_fetch_row()　　D．mysql_num_rows()

3．查询数据表中的数据使用函数（　　）。

A．mysql_query()　　B．mysql_error()

C．mysql_connect()　　D．mysql_fetch_array()

4．执行数据修改操作，使用语句（　　）。

A．select　　B．update

C．delete　　D．insert

5．mysql_fetch_row()和 mysql_fetch_array()函数的区别是（　　）。

A．mysql_fetch_row()只可以使用数字作为索引，而 mysql_fetch_array()既可以使用数字作为索引，也可以使用字符串作为索引来获取数据

B．mysql_fetch_array()只可以使用数字作为索引，而 mysql_fetch_row()既可以使用数字作为索引，也可以使用字符串作为索引来获取数据

6．下面的代码用于查询表 tb_user 中用户名字段（name）与用户输入的文本框（name）是否相同，请将下面的代码补充完整。

```
mysql_query("select * from tb_user where                    ,$conn);
```

7．将下面连接数据库的代码补充完整。

```
$conn=              ("localhost","root","111");
        ("db_database08",            );
        ("set names utf8");
```

8．获取通过 mysql_query 函数执行搜索语句获取的结果集的函数是（　　）。



9．查询数据表中的前 4 条记录，数据表的主键为 id，请将下面的查询语句补充完整。

```
mysql_query("select * from tb_user where              ",$conn);
```

10．向数据库中添加一条用户信息记录，用户名为 mr，密码为 123456，请将下面的代码补充完整。

```
mysql_query("insert into tb_user(name,pwd) values(        )",$conn);
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 6 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　避免输出中文字符串时出现乱码

将数据库中的信息显示到网页中经常会遇到这样的问题，即网页中输出的中文文字为乱码。出现这个问题的主要原因是数据库中的编码格式与页面设置的编码格式不符或者未将数据信息统一为 GBK、GB2312 或 UTF-8。本例中通过设置编码格式向用户演示如何避免输出中文字符串时出现乱码，运行结果如图 8.16 所示。



图 8.16　避免输出中文字符串时出现乱码

实战模拟 2　查询指定时间段的数据

在图书管理系统中，经常需要查询指定出版日期范围内的图书情况。如图 8.17 所示，首先在图中的文本框中输入日期范围，然后单击“查看”按钮，即可实现查找该日期范围内的所有图书信息。

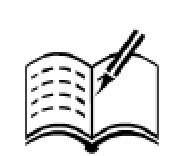


图 8.17　查询指定时间段的数据

实战模拟 3　查询从指定位置开始的 N 条记录

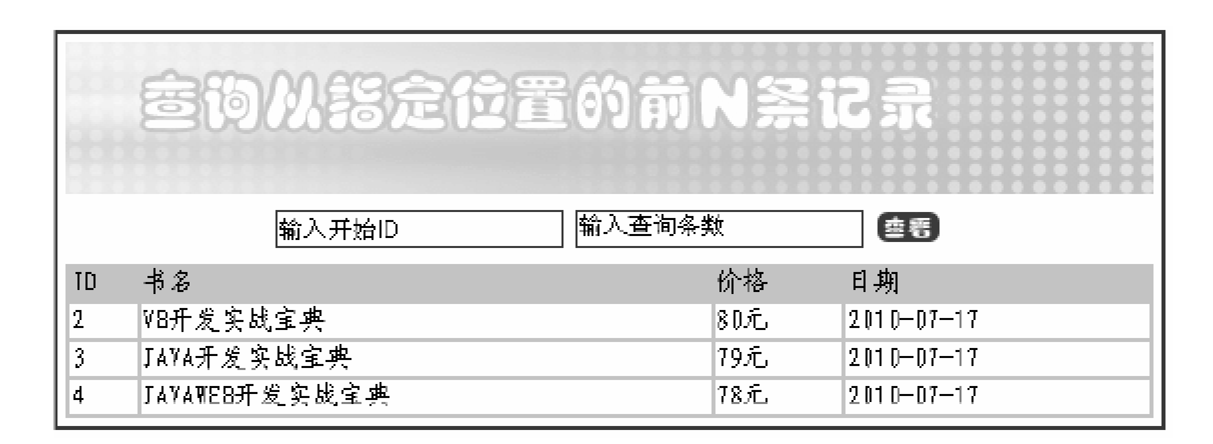
PHP 实现分页显示的原理就是从指定位置开始显示 N 条记录。本例实现查询从指定位置开







始的 N 条记录，其运行结果如图 8.18 所示。



ID	书名	价格	日期
2	VB开发实战宝典	80元	2010-07-17
3	JAVA开发实战宝典	79元	2010-07-17
4	JAVAWEB开发实战宝典	78元	2010-07-17

图 8.18　查询从指定位置开始的 N 条记录

实战模拟 4　通过 PHP 面向过程实现数据分页

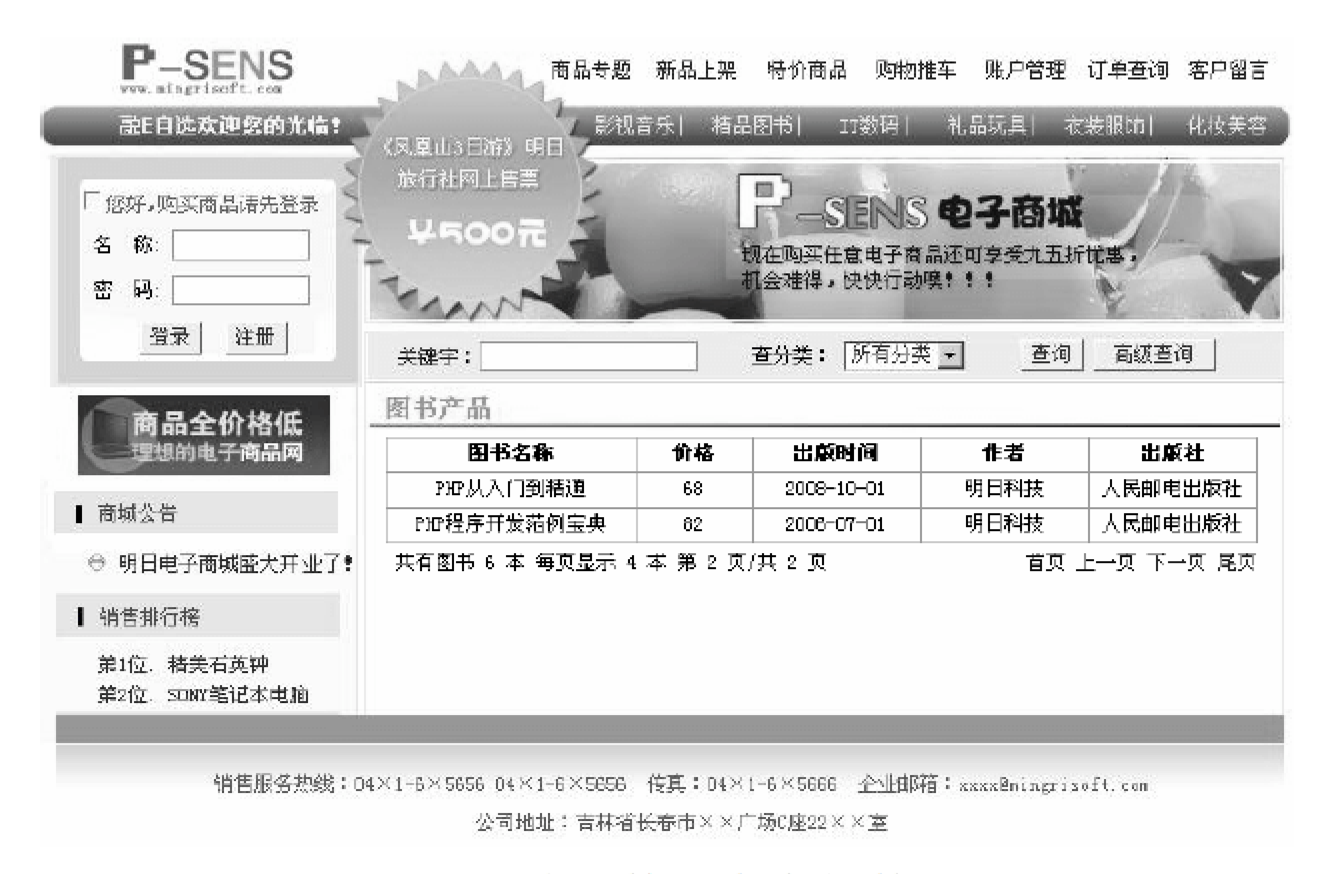
对于数据库中存储的大量的数据，如果想要进行输出，最佳的方法就是使用分页。通过分页输出数据，可以保持页面整洁，同时提高数据的浏览速度。其运行结果如图 8.19 所示。



图 8.19　分页输出数据库中的数据（一）

实战模拟 5　通过 PHP 面向对象实现数据分页

上例是通过 PHP 的面向过程的编程方式实现数据的分页输出，下面介绍通过面向对象的编程方式实现数据的分页输出。运行结果如图 8.20 所示。



图书名称	价格	出版时间	作者	出版社
PHP从入门到精通	68	2008-10-01	明日科技	人民邮电出版社
PHP程序开发范例宝典	82	2008-07-01	明日科技	人民邮电出版社

图 8.20　分页输出数据库中的数据（二）

第 2 篇

技能提高篇



第9章 字符串高级处理

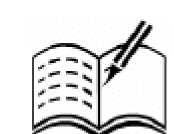
（自学视频、源程序：配套资源\mr\9\）

字符串的操作，主要是使用PHP的内置STRING函数库，通过它实现对字符串的各种操作，是Web程序开发不可缺少的内容之一。PHP程序员必须熟练地掌握字符串的处理技术，只有这样才能编写出更加实用、完善的Web程序。本章将对字符串的常用技术进行讲解。

学习摘要：

- 初识字符串
- 转义、还原字符串
- 截取字符串
- 分割、合成字符串
- 替换字符串
- 检索字符串
- 去掉字符串首尾空格和特殊字符
- 字符串与HTML转换





9.1　初识字符串

字符串是由零个或多个字符构成的一个集合。字符包含以下几种类型。

☑　数字类型。例如 1、2、3 等。

☑　字母类型。例如 a、b、c、d 等。

☑　特殊字符。例如#、$、%、^、&等。

☑　不可见字符。例如\n（换行符）、\r（回车符）、\t（Tab 字符）等。

其中，不可见字符是比较特殊的一组字符，其用来控制字符串格式化输出，在浏览器上不可见，只能看到字符串输出的结果。

例如，在下面的代码中通过 echo 语句输出一组字符串，程序代码如下：

```
<?php
    echo "PHP从入门到精通\rASP从入门到精通\nJSP程序开发范例宝典\tPHP函数参考大全"; //输出字符串
?>
```

在 IE 浏览器中不能直接查看到字符串的运行结果，只有通过“查看源文件”才能看到不可见字符串的运行结果。

运行结果为：PHP 从入门到精通
ASP 从入门到精通
JSP 程序开发范例宝典　PHP 函数参考大全

9.2　转义、还原字符串

视频讲解：配套资源\mr\9\video\转义、还原字符串.exe

在 PHP 编程的过程中经常会遇到这样的问题，即：将数据插入到数据库中时可能会引起一些问题，出现错误或乱码等，因为此时数据库将传入的数据中的字符解释成控制符。针对这样的问题，需要对特殊的字符进行转义。

因此，在 PHP 语言中提供了专门处理这些问题的技术，通过 addslashes()函数和 stripslashes()函数转义和还原字符串。

addslashes()函数用来为字符串 str 加入斜杠“\”，对指定字符串中的字符进行转义。它可以转义的字符包括：单引号“'”、双引号“"”、反斜杠“\”、NULL 字符“0”。语法如下：

```
string addslashes ( string str)
```

参数 str 为将要被操作的字符串。

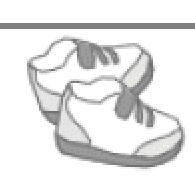
指点迷津：

addslashes()函数常用于在生成 SQL 语句时，对 SQL 语句中的部分字符进行转义。

既然有转义，就应该有还原。stripslashes()函数用于将 addslashes()函数转义后的字符串 str 还原。stripslashes()函数语法如下：

```
string stripslashes(string str);
```

参数 str 为将要被操作的字符串。



例 9.1 应用 addcslashes()函数对字符串进行转义，然后应用 stripslashes()函数进行还原，代码如下：（**实例位置：配套资源\mr\9\example\9.1**）

```
<?php
 $str = "select * from tb_book where bookname = 'PHP编程宝典'";
 $a = addslashes($str);                    //对字符串中的特殊字符进行转义
 echo $a."<br>";                           //输出转义后的字符
 $b = stripslashes($a);                    //对转义后的字符进行还原
 echo $b."<br>";                           //将字符原义输出
?>
```

运行结果为：select * from tb_book where bookname = \'PHP 编程宝典\'
select * from tb_book where bookname = 'PHP 编程宝典'

多学两招：

所有数据在插入数据库之前，都有必要应用 addslashes()函数进行字符串转义，以免特殊字符未经转义在插入数据库时出现错误。另外，对于应用 addslashes()函数实现的自动转义字符串可以应用 stripslashes()函数进行还原，但数据在插入数据库之前必须再次进行转义。

指点迷津：

如何控制转义、还原字符串的范围？

通过 addcslashes()函数和 stripcslashes()函数可以对指定范围内的字符串进行转义、还原。

（1）addcslashes()函数，对指定字符串中的字符进行转义，即在指定的字符 charlist 前加上反斜杠。通过该函数可以将要添加到数据库中的字符串进行转义，从而避免出现乱码等问题。语法如下：

```
string addcslashes ( string str, string charlist)
```

参数 str 为将要被操作的字符串；参数 charlist 指定在字符串中的哪些字符前加上反斜杠“\”。如果参数 charlist 中包含有\n、\r 等字符，则将以 C 语言风格转换，而其他非字母数字且 ASCII 码低于 32 位以及高于 126 位的字符均转换成使用 8 进制表示。

（2）stripcslashes()函数实现对 addcslashes()函数转义的字符串 str 进行还原。语法如下：

```
string stripcslashes ( string str)
```

参数 str 为将要被操作的字符串。

9.3 截取字符串

视频讲解：配套资源\mr\9\video\截取字符串.exe

在 PHP 中对字符串进行截取应用 substr()函数，该操作经常使用。

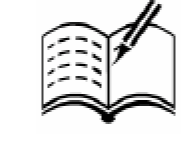
substr()函数从字符串中按照指定位置截取一定长度的字符。如果使用一个正数作为子串起点来调用这个函数，将得到从起点到字符串结束的这个字符串；如果使用一个负数作为子串起点来调用，将得到一个原字符串尾部的一个子串，字符个数等于给定负数的绝对值。语法如下：

```
string substr ( string str, int start [, int length])
```

参数 str 用来指定字符串对象；参数 start 用来指定开始截取字符串的位置，如果 start 为负数，则从字符串的末尾开始截取；参数 length，可选参数，指定截取字符的个数，如果 length







为负数，则表示取到倒数第 length 个字符。

substr()函数的操作流程如图 9.1 所示。

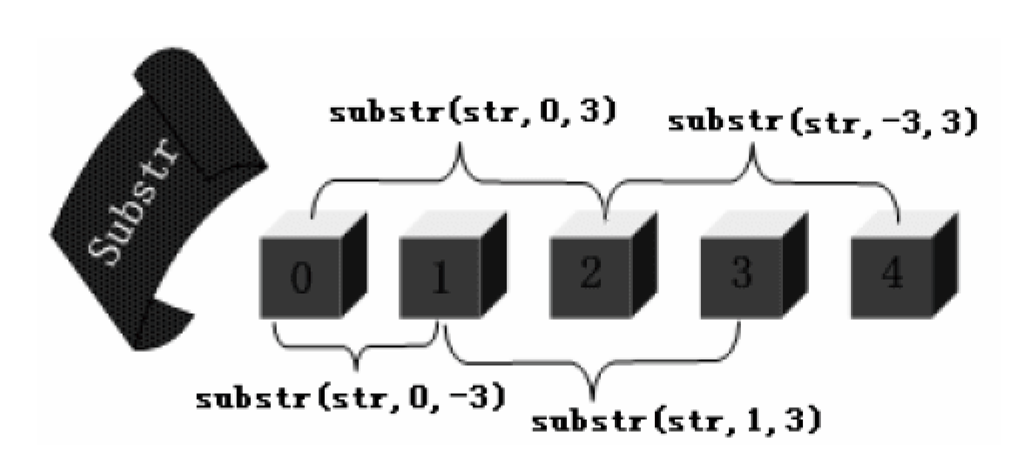


图 9.1　substr()函数的操作流程

脚下留神：

substr()函数中参数 start 的指定位置是从 0 开始计算的，即字符串中的第一个字符表示为 0。

例 9.2　在开发 Web 程序时，为了保持整个页面的合理布局，经常需要对一些（例如公告标题、公告内容、文章的标题、文章的内容等）超长输出的字符串内容进行截取，并通过“…”代替省略内容，代码如下：**（实例位置：配套资源\mr\9\example\9.2）**

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>截取字符串</title>
</head>
<body>
<?php
$str="为进一步丰富编程词典的内容和观赏性，公司决定组织“春季盎然杯”摄影大赛，本次参赛作品要求全部为春季拍摄，旨在展示我国北方地区春季生机盎然的景色。";
if(strlen($str)>40){                    //如果文本的字符串长度大于40
    echo substr($str,0,40)."…";         //输出文本的前50个字符串，然后输出省略号
}else{                                  //如果文本的字符串长度小于40
    echo $str;                          //直接输出文本
}
?>
</body>
</html>
```

运行结果如图 9.2 所示。

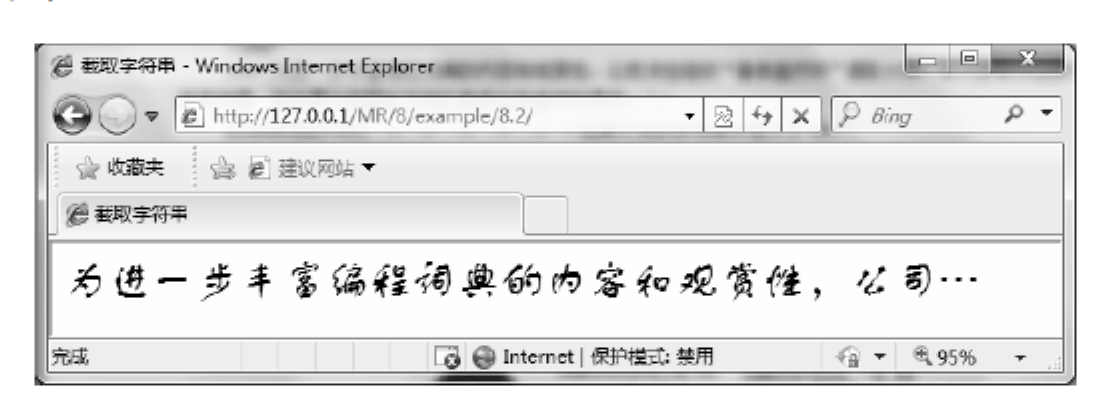
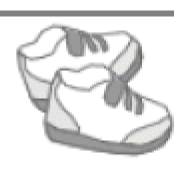
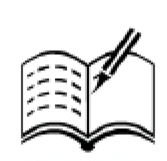


图 9.2　substr()函数截取字符串

多学两招：

在应用 substr()函数对字符串进行截取时，应该注意页面的编码格式，切记页面编码格式不能设置为 UTF-8。如果页面是 UTF-8 编码格式，那么应该使用 iconv_substr()函数进行截取。





Note

指点迷津：

strlen()函数获取字符串的长度，汉字占两个字符，数字、英文、小数点、下划线和空格各占一个字符。

另外，通过 strlen()函数还可以检测字符串长度。例如，在用户注册中，通过 strlen()函数获取用户填写用户名的长度，然后判断用户名长度是否符合指定的标准。关键代码如下：

```
<?php
if(strlen($_POST["pwd"])<6){                //检测用户密码的长度是否小于 6，弹出警告信息
    echo "<script>alert('用户密码的长度不得少于 6 位!请重新输入'); history.back();</script>";
}else{                                       //若用户密码大于等于 6 位，则弹出该提示信息
    echo "用户信息输入合法！ ";
}
?>
```

上机演练

上机演练 1　统计帖子标题的长度

统计字符串长度一般是为了给其他函数的应用做好铺垫。本例通过 strlen()函数获取帖子标题长度，运行结果如图 9.3 所示。

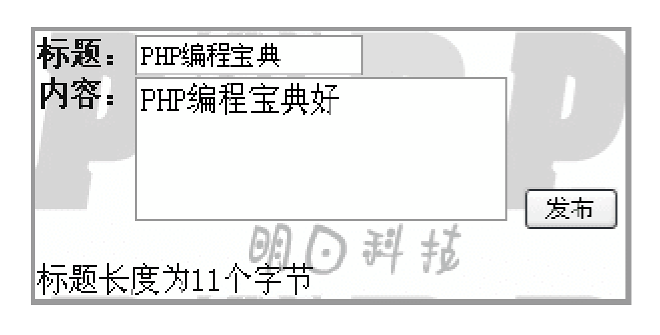


图 9.3　统计帖子标题的长度

9.4　分割、合成字符串

视频讲解：配套资源\mr\9\video\分割、合成字符串.exe

分割字符串可将指定字符串中的内容按照某个规则进行分类存储，进而实现更多的功能。例如，在电子商务网站的购物车中，可以通过特殊标识符“@”将购买的多种商品组合成一个字符串存储在数据表中，在显示购物车中的商品时，通过以“@”作为分割的标识符进行拆分，将商品字符串分割成 N 个数组元素，最后通过 for 循环语句输出数组元素，即输出购买的商品。

字符串的分割使用 explode()函数，按照指定的规则对一个字符串进行分割，返回值为数组。语法如下：

```
array explode(string separator,string str,[int limit])
```

explode()函数的参数说明如表 9.1 所示。

表 9.1　explode()函数的参数说明

参　　数	说　　明
separator	必选参数，指定的分割符。如果 separator 为空字符串（""），则返回 False；如果 separator 所包含的值在 str 中无法找到，那么返回包含 str 单个元素的数组





续表

参　　数	说　　明
str	必选参数，指定将要被进行分割的字符串
limit	可选参数，如果设置了 limit 参数，则返回的数组中包含最多 limit 个元素，而最后的元素将包含 string 的剩余部分；如果 limit 参数是负数，则返回除了最后的-limit 个元素外的所有元素

例 9.3　应用 explode()函数对指定的字符串以@为分隔符进行拆分，并输出返回的数组，代码如下：（**实例位置：配套资源\mr\9\example\9.3**）

```
<?php
    $str="PHP编程宝典@NET编程宝典@ASP编程宝典@JSP编程宝典";      //定义字符串常量
    $str_arr=explode("@",$str);                                //应用标识@分割字符串
    print_r($str_arr);                                         //输出字符串分割后的结果
?>
```

运行结果为：Array ([0] => PHP 编程宝典 [1] => NET 编程宝典 [2] => ASP 编程宝典 [3] => JSP 编程宝典)

多学两招：

如何将数组中的元素合成字符串？

既然可以对字符串进行分割，返回数组，那么就一定可以对数组进行合成，返回一个字符串，这就是 implode()函数。下面将数组中的元素组合成一个新字符串。语法如下：

string implode(string glue, array pieces)

参数 glue 是字符串类型，指定分隔符；参数 pieces 是数组类型，指定要被合并的数组。

例如，应用 implode()函数将数组中的内容以*为分隔符进行连接，从而组合成一个新的字符串，代码如下：

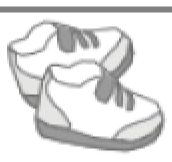
```
<?php
$str="PHP编程宝典*NET编程宝典*ASP编程宝典*JSP编程宝典";   //定义字符串常量
$str_arr=explode("*",$str);                              //应用标识*分割字符串
$array=implode("*",$str_arr);                            //将数组合成字符串
echo $array;                                             //输出字符串
?>
```

运行结果为：PHP 编程宝典*NET 编程宝典*ASP 编程宝典*JSP 编程宝典

上机演练

上机演练 2　购物车中数据的读取

SESSION 购物车的设计原理：当用户登录网站时，系统将为每个用户分配两个 SESSION 变量：$producelist 和$quatity，分别用来存储用户放入购物车中的商品 id 和商品数量，将@作为分隔符，实现将多个 id 值同时保存在一个$producelist 变量中。例如，用户分别将 id 为 1、2、3 的商品放入购物车中，这时 SESSION 变量$producelist 的值应该为“1@2@3@”，这就是 SESSION 购物车的设计原理。读取购物车中商品信息的运行结果如图 9.4 所示。




Note



图 9.4　购物车中数据的读取

9.5　替换字符串

视频讲解：配套资源\mr\9\video\替换字符串.exe

利用字符串的替换技术可以屏蔽帖子或留言板中的非法字符，也可以对查询的关键字进行描红。PHP 中提供 str_ireplace()函数和 substr_replace()函数实现字符串的替换功能。

9.5.1　str_ireplace()函数

str_ireplace()函数使用新的子字符串（子串）替换原始字符串中被指定要替换的字符串。语法如下：

```
mixed str_ireplace ( mixed search, mixed replace, mixed subject [, int &count])
```

将所有在参数 subject 中出现的参数 search 以参数 replace 取代，参数&count 表示取代字符串执行的次数。

str_ireplace()函数的参数说明如表 9.2 所示。

表 9.2　str_ireplace()函数的参数说明

参　　数	说　　明
search	必选参数，指定需要查找的字符串
replace	必选参数，指定替换的值
subject	必选参数，指定查找的范围
count	可选参数，获取执行替换的数量

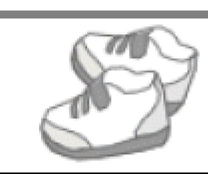
例 9.4　应用 str_ireplace()函数将文本中的字符串“mrsoft”替换为“吉林省明日科技”，代码如下：（**实例位置：配套资源\mr\9\example\9.4**）

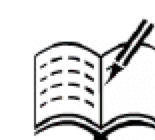
```
<?php
$str="MRSOFT公司是一家以计算机软件技术为核心的高科技企业";        //定义字符串常量
echo str_ireplace("mrsoft","吉林省明日科技",$str);              //输出替换后的字符串
?>
```

运行结果为：吉林省明日科技公司是一家以计算机软件技术为核心的高科技企业。

指点迷津：

本函数不区分大小写。如果需要对大小写加以区分，可以使用 str_replace()函数。





多学两招：

关键字描红是指将查询关键字以特殊的颜色、字号或字体进行标识，这样可以使浏览者快速检索到所需的关键字，方便浏览者从搜索结果中查找所需内容。查询关键字描红适用于模糊查询。

9.5.2　substr_replace()函数

substr_replace()函数对指定字符串中的部分字符串进行替换。语法如下：

```
string substr_replace(string str,string repl,int start,[int length])
```

substr_replace()函数的参数说明如表 9.3 所示。

表 9.3　substr_replace()函数的参数说明

参　　数	说　　明
str	指定要操作的原始字符串
repl	指定替换后的新字符串
start	指定替换字符串开始的位置。正数表示起始位置从字符串开头开始；负数表示起始位置从字符串的结尾开始；0 表示起始位置从字符串中的第一个字符开始
length	可选参数，指定返回的字符串长度，默认值是整个字符串。正数表示起始位置从字符串的开头开始；负数表示起始位置从字符串的结尾开始；0 表示“插入”非“替代”

脚下留神：

如果参数 start 设置为负数，而参数 length 数值小于或等于 start 数值，那么 length 的值自动为 0。

例 9.5　使用 substr_replace()函数对指定字符串进行替换，代码如下：**（实例位置：配套资源\mr\9\example\9.5）**

```
<?php
$str="用今日的辛勤工作，换明日的双倍回报！";                //定义字符串常量
$replace="百倍";                                          //定义要替换的字符串
echo substr_replace($str,$replace,26,4);                  //替换字符串
?>
```

在上面的代码中，使用 substr_replace()函数将字符串“双倍”替换为字符串“百倍”。

运行结果为：用今日的辛勤工作，换明日的百倍回报！

上机演练

上机演练 3　查询关键字描红

查询关键字描红在百度、谷歌等搜索引擎中是非常实用的，通过它可以凸显出要查找的内容。本例通过 str_ireplace()函数实现查询关键字描红的功能，其运行结果如图 9.5 所示。





Note

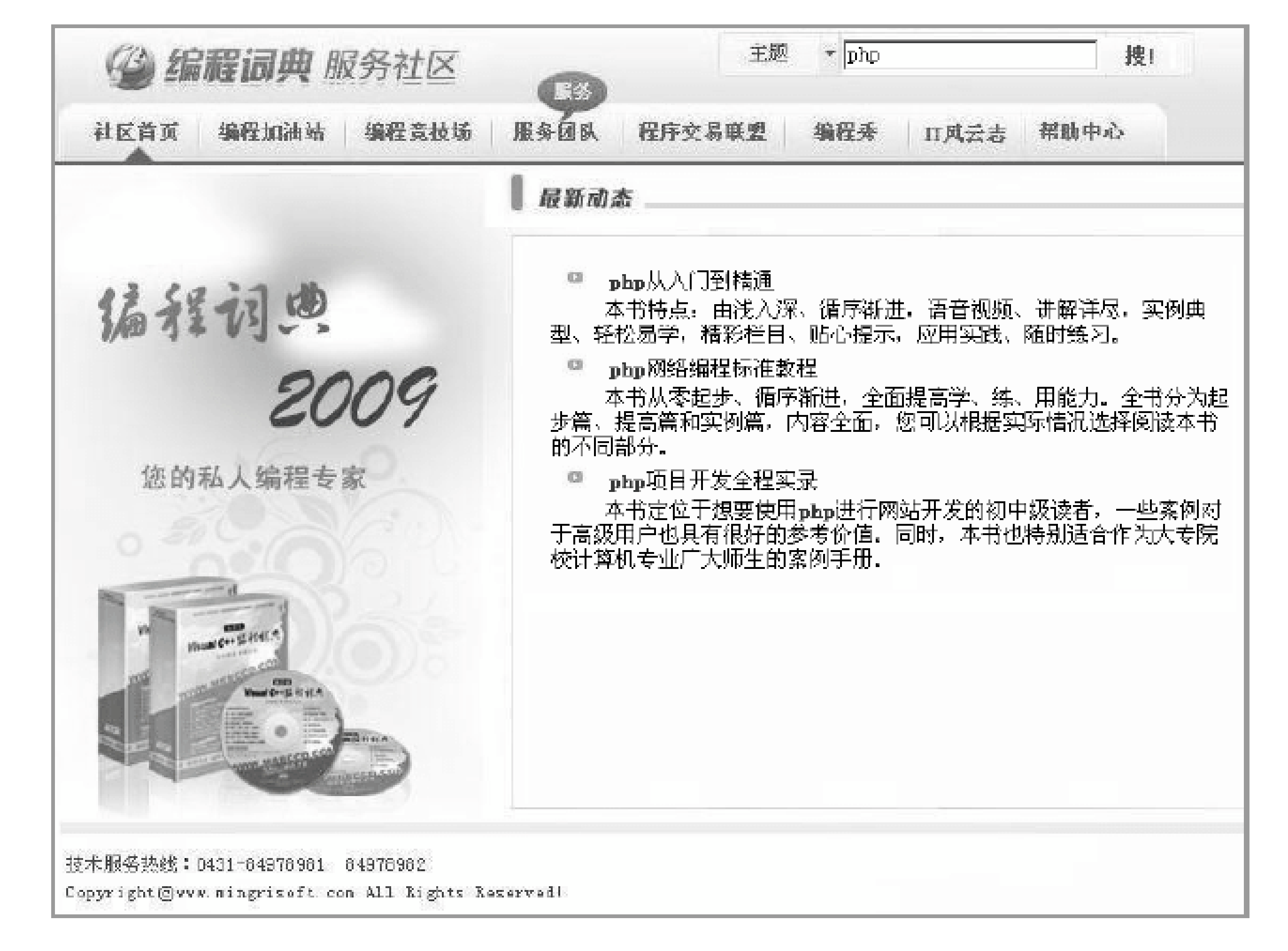


图 9.5　查询关键字描红

9.6　检索字符串

视频讲解：配套资源\mr\9\video\检索字符串.exe

在 PHP 中，提供了很多应用于字符串查找的函数，如 strstr()函数和 substr_count()函数。PHP 也可以像 Word 那样实现对字符串的查找功能。

9.6.1　strstr()函数

strstr()函数获取一个指定字符串在另一个字符串中首次出现的位置到后者末尾的子字符串。如果执行成功，则返回剩余字符串（存在相匹配的字符）；否则返回 False。语法如下：

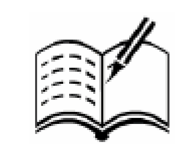
```
string strstr ( string haystack, string needle)
```

参数 haystack 指定从哪个字符串中进行搜索；参数 needle 指定搜索的对象。如果该参数是一个数值，那么将搜索与这个数值的 ASCII 码相匹配的字符。

例 9.6　应用 strstr()函数获取上传图片的后缀，并判断上传图片格式是否正确，如果正确则将图片上传到服务器根目录下的 upload 文件夹下，否则给出提示信息，代码如下：**（实例位置：配套资源\mr\9\example\9.6）**

```
<form name="form" method="post" action="index.php" enctype="multipart/form-data">
    <input name="u_file" type="file" size="24"/>
    （<span class="STYLE1">*上传图片是.jpg格式，大小不能超过1.2MB</span>）
    <input type="image" name="imageField" src="imges/sc.bmp" onClick="form.submit();">
</form>
<?php
header("Content-type:text/html;charset=utf-8");
if($_FILES[u_file][name]==true){
    $file_path = "./upload\\";                          //定义图片在服务器中的存储位置
    $picture_name=$_FILES[u_file][name];          //获取上传图片的名称
    $picture_name=strstr($picture_name , ".");  //通过strstr()函数截取上传图片的后缀
    if($picture_name!= ".jpg" && $picture_name!= ".JPG" ){          //根据后缀判断图片的格式
```





```
                echo "<script>alert('上传图片格式不正确,请重新上传'); window.location.href='index.php';</script>";
        }else if($_FILES[u_file][tmp_name]){
                move_uploaded_file($_FILES[u_file][tmp_name],$file_path.$_FILES[u_file][name]);//执行图片上传
                echo "<script>alert('图片上传成功!'); window.location.href='index.php';</script>";
        }else{
                echo "<script>alert('上传图片失败！'); window.location.href='index.php';</script>";
        }
    }
    ?>
```

运行结果如图 9.6 所示。



图 9.6　应用 strstr()检索上传图片的后缀

多学两招：

检索字符串函数扩展

strstr()函数区分大小写，如果不需要对大小写加以区分，则可以使用 stristr()函数。

strstr()函数从指定字符在另一个字符串中首次出现的位置开始查找；如果想从指定字符在另一个字符串中最后一次出现的位置开始查找，则可以使用 strrchr()函数。strrchr()函数区分大小写。

stripos()函数查找指定字符串（A）在另一个字符串（B）中首次出现的位置。该函数不区分大小写。如果要区分大小写，则可以使用 strpos()函数。

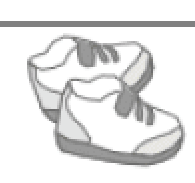
strripos()函数查找指定字符串（A）在另一个字符串（B）中最后一次出现的位置。该函数不区分大小写。如果要区分大小写，则可以使用 strrpos()函数。

9.6.2　substr_count()函数

检索字符串的函数，都是检索指定字符串在另一字符串中出现的位置，这里再介绍一个检索子串在字符串中出现次数的函数——substr_count()函数。该函数获取子串在字符串中出现的次数。语法如下：

```
int substr_count(string haystack,string needle)
```

参数 haystack 是指定的字符串；参数 needle 是指定的子串。



例如，使用 substr_count()函数获取子串在字符串中出现的次数，代码如下：

```
<?php
$str="PHP编程宝典、JavaWeb编程宝典、Java编程宝典、VB编程宝典"; //输出查询的字符串
echo substr_count($str,"编程宝典");                          //输出查询的字符串
?>
```

运行结果为：4

指点迷津：

检索子串出现的次数一般常用于搜索引擎中，针对子串在字符串中出现的次数进行统计，便于用户第一时间掌握子串在字符串中出现的次数。

上机演练

上机演练 4　统计查询关键字的出现次数

在站内搜索中我们往往是要列出符合条件的关键字有多少个。而本例通过字符串函数 substr_count()统计查询关键字的出现次数，运行结果如图 9.7 所示。

被搜索的文本：PHP作为全球最普及、应用最广泛的互联网开发语言之一，从1994年诞生至今已被2000多万个动态网站采用，全球知名互联网公司Google、Yahoo、eBay和中国知名网站新浪、百度、阿里巴巴等均采用PHP技术！
关键字： 搜索
PHP作为全球最普及、应用最广泛的互联网开发语言之一，从1994年诞生至今已被2000多万个动态网站采用，全球知名互联网公司Google、Yahoo、eBay和中国知名网站新浪、百度、阿里巴巴等均采用PHP技术！
关键字出现2次。

图 9.7　统计查询关键字的出现次数

9.7　去掉字符串首尾空格和特殊字符

用户在输入数据时经常会在无意中输入多余的空白字符，在某些情况下，字符串中不允许出现空白字符和特殊字符，这就需要去除字符串中的空白字符和特殊字符。在 PHP 中提供了 trim()函数去除字符串左右两边的空白字符和特殊字符、ltrim()函数去除字符串左边的空白字符和特殊字符、rtrim()函数去除字符串中右边的空白字符和特殊字符。

9.7.1　ltrim()函数

ltrim()函数用于去除字符串左边的空白字符或指定字符串。语法如下：

```
string ltrim( string str [,string charlist]);
```

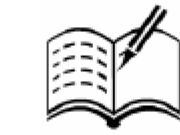
参数 str 是要操作的字符串对象；参数 charlist 为可选参数，指定需要从指定的字符串中删除哪些字符，如果不设置该参数，则所有的可选字符都将被删除。ltrim()函数的参数 charlist 的可选值如表 9.4 所示。

表 9.4　ltrim()函数的参数 charlist 的可选值

参　数　值	说　　明
\0	NULL，空值
\t	tab，制表符

Note





续表

参　数　值	说　　明
\n	换行符
\x0B	垂直制表符
\r	回车符
" "	空白字符

指点迷津：

除了以上默认的过滤字符列表外，也可以在 charlist 参数中提供要过滤的特殊字符。

例 9.7 使用 ltrim()函数去除字符串左边的空白字符及特殊字符“(:*_*”，代码如下：**（实例位置：配套资源\mr\9\example\9.7）**

```
<?php
$str="   (:@_@   有一条路走过了总会想起！   @_@:)    ";
$strs="   (:*_*   有一条路走过了总会想起！   *_*:)    ";
echo $str."\n";                         //输出原始字符串
echo ltrim($str)."\n";                  //去除字符串左边的空白字符
echo $strs."\n";                        //输出原始字符串
echo ltrim($strs,"(:*_* ");             //去除字符串左边的特殊字符(:*_*
?>
```

运行结果如图 9.8 所示。

9.7.2 rtrim()函数

rtrim()函数用于去除字符串右边的空白字符和特殊字符。语法如下：

```
string rtrim(string str [,string charlist]);
```

参数 str 指定操作的字符串对象；参数 charlist 为可选参数，指定需要从指定的字符串中删除哪些字符，如果不设置该参数，则所有的可选字符都将被删除。参数 charlist 的可选值如表 9.4 所示。

例如，使用 rtrim()函数去除字符串右边的空白字符及特殊字符“(:*_*”，代码如下：

```
<?php
$str="   (:@_@   有一条路走过了总会想起！   @_@:)    ";
$strs="   (:*_*   有一条路走过了总会想起！   *_*:)    ";
echo $str."\n";                         //输出原始字符串
echo rtrim($str)."\n";                  //去除字符串右边的空白字符
echo $strs."\n";                        //输出原始字符串
echo rtrim($strs,"   *_*:)");           //去除字符串右边的特殊字符(:*_*
?>
```

运行结果如图 9.9 所示。

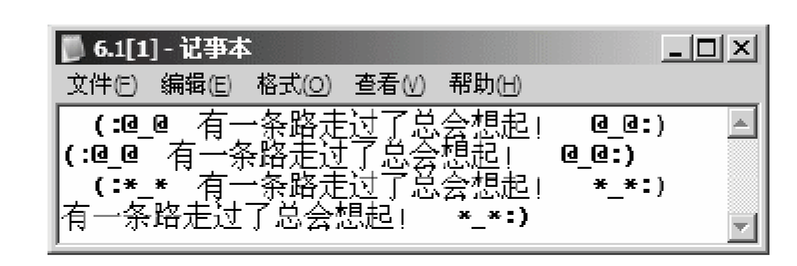


图 9.8　去除字符串左边的空白字符及特殊字符

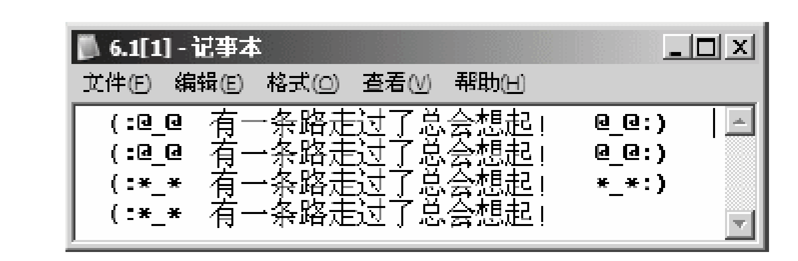
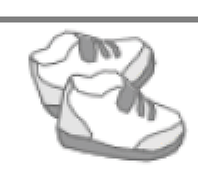


图 9.9　去除字符串右边的空白字符及特殊字符



9.7.3 trim()函数



Note

trim()函数用于去除字符串开始位置和结束位置的空白字符，并返回去掉空白字符后的字符串。语法如下：

```
string trim(string str [,string charlist]);
```

参数 str 是操作的字符串对象；参数 charlist 为可选参数，指定需要从指定的字符串中删除哪些字符，如果不设置该参数，则所有的可选字符都将被删除。参数 charlist 的可选值如表 9.4 所示。

例如，使用 trim()函数去除字符串左右两边的空白字符及特殊字符“\r\r(: :)”。其代码如下：

```
<?php
$str="    \r\r(:@_@去除字符串左右两边的空白和特殊字符 @_@:)        ";
echo $str."\n";                          //输出原始字符串
echo trim($str)."\n";                    //去除字符串左右两边的空白字符
echo trim($str,"\r\r(:@_@ @_@:)");       //去除字符串左右两边的空白字符和特殊字符\r\r(:@_@ @_@:)
?>
```

运行结果如图 9.10 所示。

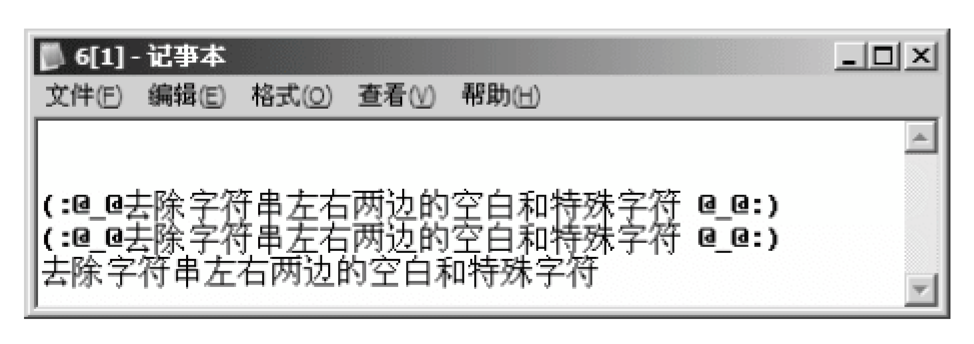


图 9.10 去除字符串左右的空格和特殊字符

上机演练

上机演练 5 去除帖子标题的首尾空格

用户在编写代码时也许会在不经意间多了一个或多个空格，使得程序无论如何调试都无法正常运行。本例通过 trim()函数去除帖子标题的首尾空格，运行结果如图 9.11 所示。

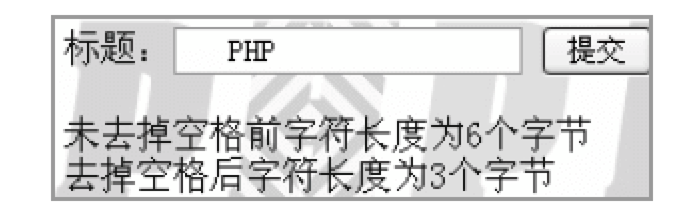


图 9.11 去除帖子标题的首尾空格

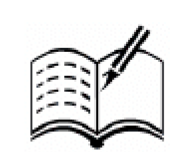
9.8 字符串与 HTML 转换

字符串与 HTML 之间的转换直接将源代码在网页中输出，而不被执行。该操作应用最多的地方就是在论坛或博客的帖子输出中，通过转换直接将提交的源码输出，而确保源码不被解析。完成这个操作主要应用 htsdmlentities()函数。

htmlentities()函数将所有的字符都转成 HTML 字符串，语法如下：

```
string htmlentities(string string,[int quote_style],[string charset])
```





htmlentities()函数的参数说明如表 9.5 所示。

表 9.5　htmlentities()函数的参数说明

参　数	说　明
string	必选参数，指定要转换的字符串
quote_style	可选参数，选择如何处理字符串中的引号，有 3 个可选值：（1）ENT_COMPAT，转换双引号，忽略单引号，它是默认值；（2）ENT_NOQUOTES，忽略双引号和单引号；（3）ENT_QUOTES，转换双引号和单引号
charset	可选参数，确定转换所使用的字符集，默认字符集是“ISO-8859-1”，指定字符集后就能够避免转换中文字符出现乱码的问题

htmllentities()函数支持的字符集如表 9.6 所示。

表 9.6　htmlentities()函数支持的字符集

字　符　集	说　明
BIG5	繁体中文
BIG5-HKSCS	香港扩展的 BIG5，繁体中文
cp866	DOS 特有的西里尔（Cyrillic）字符集
cp1251	Windows 特有的西里尔字符集
cp1252	Windows 特有的西欧字符集
EUC-JP	日文
GB2312	简体中文
ISO-8859-1	西欧，Latin-1
ISO-8859-15	西欧，Latin-9
KOI8-R	俄语
Shift-JIS	日文
UTF-8	ASCII 兼容的多字节 8 编码

例 9.8　使用 htmlentities()函数将论坛中的帖子进行输出，然后将转换后的代码和未转换的代码进行对比，代码如下：**（实例位置：配套资源\mr\9\example\9.8）**

```
<?php
$str='<table width="300" border="1" cellpadding="1" cellspacing="1" bgcolor="#0198FF">
    <tr>
    <td align="center" height="35"   bgcolor="#FFFFFF">明日科技——用今日的辛勤工作，换明日百倍回报！ </td>
      </tr>
      <tr>
        <td align="center" bgcolor="#FFFFFF" ><img src="images/beg.JPG"></td>
      </tr>
    </table>';
    echo htmlentities($str,ENT_QUOTES," utf-8")."<br>";          //设置转换的字符集为UTF-8
?>
```

运行结果如图 9.12 所示。



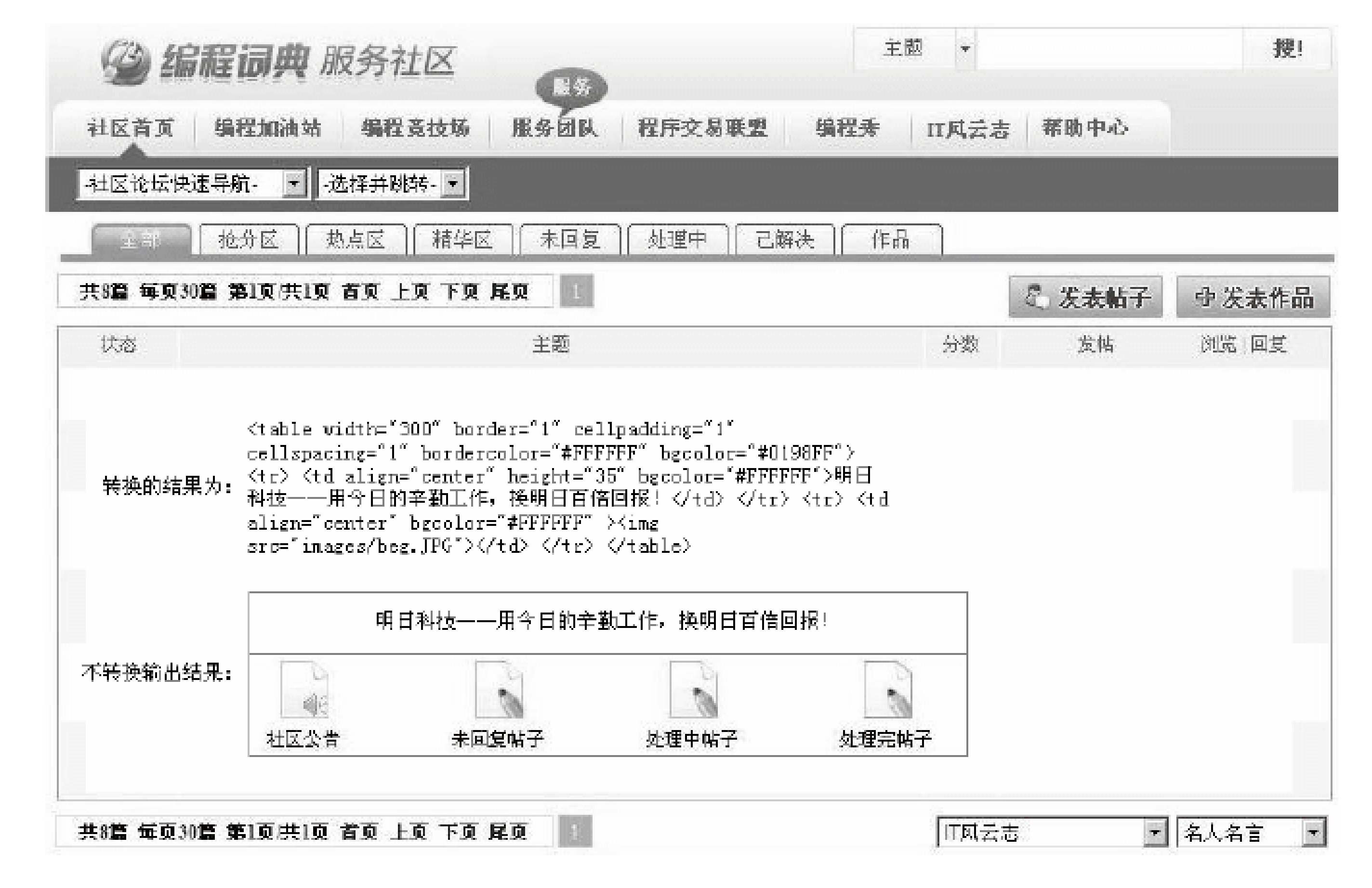



图 9.12　字符串与 HTML 转换结果的对比

多学两招：

正则表达式是一种描述字符串结构的语法规则，是一个特定的格式化模式，可以匹配、替换、截取匹配的字串。对于用户来说，在注册某个网站的会员时都提交过邮箱、电话号码或邮编等注册信息，但是当填写的注册信息的格式不正确时就会弹出一个提示信息，提示填写的邮箱或电话号码的格式不正确，那么这个判断的依据就是正则表达式。通过正则表达式来对提交的信息进行匹配，如果不匹配则给出提示信息。

例如，匹配邮箱和电话号码格式的正则表达式如下：

/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/　　　　//验证邮箱格式是否正确

/^(\d{3}-)(\d{8})$|^(\d{4}-)(\d{7})$|^(\d{4}-)(\d{8})$/;　　　　//验证座机号码

如果要应用正则表达式，则必须将其放置到某种语言中，通过语言中的方法调用正则表达式对提交的信息进行验证。在 PHP 中有两套函数库支持正则表达式的处理操作。一套是由 PCRE（Perl Compatible Regular Expression）库提供的，与 Perl 语言兼容的正则表达式函数，其以"preg_"为前缀命名函数，而且表达式要包含在定界符"/"之内；另一套是 POSIX（Portable Operating System Interface）扩展语法正则表达式函数，其以"ereg_"为前缀命名函数。两套函数库的功能非常相似，但是在执行效率上 PCRE 函数库要略优于 POSIX 函数库。

指点迷津：

一个完整的正则表达式由两部分构成，即元字符和文本字符。元字符是具有特殊含义的字符，例如前面提到的"*"和"？"；文本字符就是普通的文本，如字母和数字等。PCRE 风格的正则表达式一般都放置在定界符"/"中间，如"/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/"、"/^http:\/\/(www\.)?.+.?$/"。

为了便于读者理解，除了个别字符外，将元字符的相关说明和分类归纳在表 9.7 中。

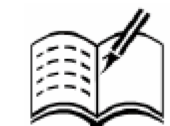
表 9.7　元字符的分类

元　字　符	说　　明	举　　例
行定位符（^和$）	行定位符就是用来描述字串的边界	^tm
单词定界符（\b、\B）	表达式 tm 可以对字符串中任意位置出现的字串 tm 进行匹配	\btm\b



续表

元　字　符	说　　明	举　　例
字符类（[ ]）	方括号内不区分大小写	[Tt][Mm]
选择字符（|）	该字符可以理解为“或”，与“[ ]”相同	(T|t)(M|m)
连字符（-）	变量的命名规则是只能以字母和下划线开头	[a-zA-Z]
排除字符（[^]）	匹配不符合命名规则的变量	[^a-zA-Z]
限定符（? * + {n,m}）	对于重复出现字母或字串，可以使用限定符来实现匹配	go{2,20}gle
点号字符（.）	匹配除换行符外的任意一个字符	^s.t$
转义字符（\）	正则表达式中的转移字符（\）和 PHP 中的大同小异，都是将特殊字符转义为普通字符	[0-9]{1,3}(\.[0-9]{1,3}){3}
括号字符（()）	改变限定符的作用范围	(thir|four)th



上机演练

上机演练 6　验证电话号码的格式是否正确

在表单注册时往往要求用户书写座机电话号码，而座机电话号码是由 12 位或 11 位数字组成的，所以一定要对电话号码的位数和格式进行限制。通过正则表达式和正则表达式函数 preg_match()对电话号码格式进行验证，运行结果如图 9.13 所示。

上机演练 7　验证 E-mail 地址格式是否正确

互联网发展到了今天，几乎所有的 Web 爱好者都有了自己的 E-mail 地址，无论用户申请的是 126 邮箱还是 163 邮箱，E-mail 地址的格式是固定的。本例通过 preg_match()正则匹配函数和正则表达式验证 E-mail 地址格式是否正确，运行结果如图 9.14 所示。

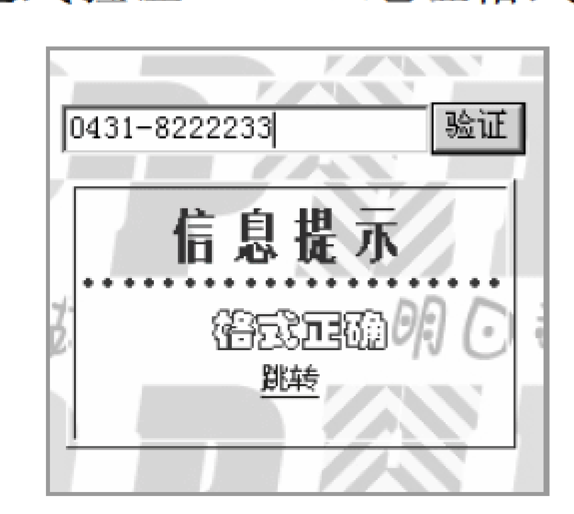


图 9.13　验证电话号码的格式

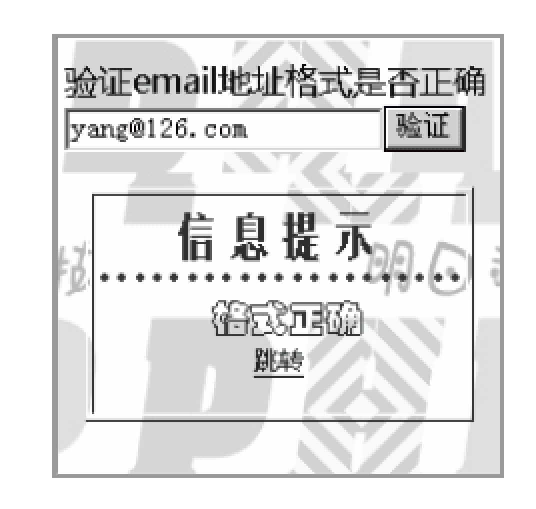
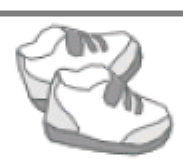
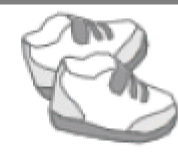


图 9.14　验证 E-mail 地址格式是否正确

本 章 摘 要

1．字符串的转义、还原技术。
2．字符串的截取技术。
3．字符串的分割、合成方法。
4．字符串的检索、替换方法。
5．字符串中空格和特殊字符串的去除技术。
6．字符串与 HTML 的转换技术。
7．应用正则表达式与正则表达式函数对电话号码和 E-mail 地址进行验证。


习　　题

1．下列哪个函数是将数组转换为字符串（　　）。
   A．imploed()　　B．explode()　　C．arsort()　　D．natsort()
2．将字符串中所有英文单字的开头字母转换为大写的函数应该是（　　）。
   A．ucfirst()　　B．strtolower()　　C．strtpupper()　　D．ucwords()
3．以下字符的长度是（　　）。

```
<?php
 $text="   \tllo   ";
 Echo strlen(trim($text));
?>
```

   A．9　　B．5　　C．7　　D．3
4．PHP 中，下列哪个函数是将字符串前后颠倒（　　）。
   A．strrev()　　B．strrpos()　　C．strstr()　　D．strfirst()
5．下面哪个正则表达式可以从服务器上取到访问域名第二层（　　）。
   A．preg_replace（"/.*([^\.\/]+)(\..(com|net|org))?\.[^\.\/]+$/","\1",$_SERVER['HTTP_HOST']）
   B．preg_replace（"/.*?([^\.\/]+)(\..(com|net|org))?\.[^\.\/]+$/","\1",$_SERVER['HTTP_HOST']）
   C．preg_replace（"/.*([^\.\/]+)(\..(com|net|org))?\.[^\.\/]+$/","\1",$HTTP_HOST）
   D．preg_replace（"/.*?([^\.\/]+)(\..(com|net|org))?\.[^\.\/]+$/","\1",$_SERVER['HTTP_HOST']）
6．在横线处填写使用的函数。

```
<?php
      $email='mingfisoft@mingrisott.com';
      $str=___①__($email,'@');
      $info=___②__('.',$str);
      ___③_（$info）;
?>
```

输出结果为：Array（[0]=>@mingrisoft[1]=>com[2]=>cn）
7．以下代码运行的结果为（　　）。

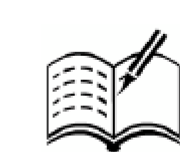
```
<?php
      $first="This course is very easy!";
      $second=explode("",$first);
      $first=implode(",",$second);
      echo $first;
?>
```

8．PHP 同时支持两种风格的正则表达式，它们分别是（　　）和（　　）。
9．在 PHP 中，定义字符串需要使用特殊符号（　　）。
10．在 PHP 中，验证 IP 地址需要使用函数（　　）。

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和





实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 7 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　超长文本的分页输出

超长文本的分页输出需要使用 3 个方面的技术：第一方面，自定义函数，通过自定义函数读取文本文件，可以避免中文字符串出现乱码；第二方面，字符串函数，需要通过 strlen()函数计算字符串的长度，通过 substr()函数对字符串进行截取；第三方面，文件系统函数，通过 file_get_contents()函数读取文本文件中的数据。超长文本分页输出的运行结果如图 9.15 所示。



图 9.15　超长文本的分页输出

实战模拟 2　货币数据的格式化输出

货币数据不同于整型数据，它是存在一定格式的。本例通过 number_format()函数实现金额的格式化输出，运行结果如图 9.16 所示。

实战模拟 3　统一英文注册用户首字母的大小写

网站的国际化已经不是一个炙手可热的话题，但是由于地区差异，在使用英文填写注册信息时首字母大小写得不到统一。运用 ucfirst()函数实现英文注册用户首字母统一为大写是一个不错的选择，其运行结果如图 9.17 所示。

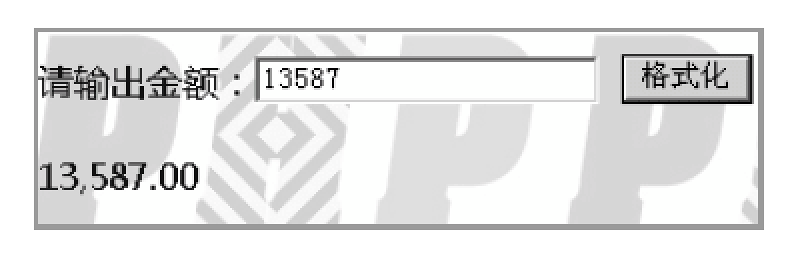


图 9.16　金额的格式化输出



图 9.17　统一英文注册用户首字母为大写

实战模拟 4　解决用 substr()函数对中文字符串截取时乱码的问题

substr()函数是按字节进行截取字符串的，在截取中文字符串时，由于一个汉字由两个字符组成，如果只截取 1 个字符就会出现乱码。使用自定义函数可以解决对中文字符串截取时乱码的问题，运行结果如图 9.18 所示。

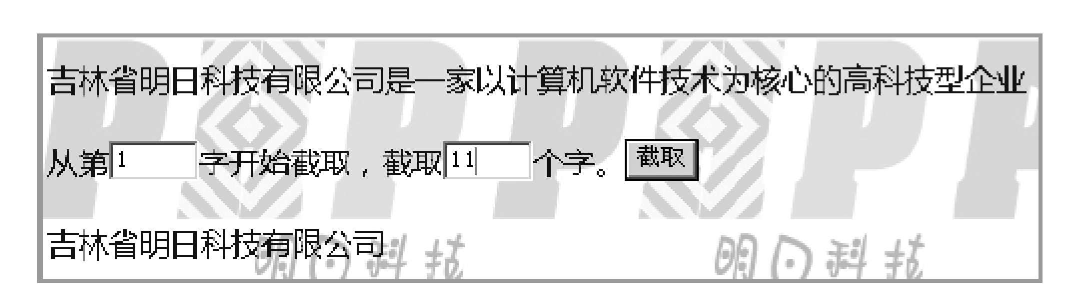


图 9.18　解决用 substr()函数对中文字符串截取时乱码的问题

第10章

日期和时间处理

（自学视频、源程序：配套资源\mr\10\）

在程序设计中，日期和时间是非常重要的，通过日期和时间可以记录数据库中数据的处理时间、计划的预计完成时间、系统当前时间、文件的操作时间、网站在某个时间段的访问量、系统管理员的登录时间以及网站遭到非法入侵的时间等。本章将对日期和时间的处理技术进行详细讲解。

学习摘要：

- PHP 的时间观念
- 通过 date_default_timezone_set()函数设置网站当前时区
- UNIX 时间戳
- 日期和时间处理





10.1　PHP 的时间观念

视频讲解：配套资源\mr\10\video\PHP 的时间观念.exe

在 PHP 语言中，日期、时间函数依赖于服务器的地区设置，而 PHP 默认设置的是标准的格林威治时间（即采用的是零时区）。如果没有对 PHP 的时区进行设置，并且用户的当地时间是北京时间，那么通过 PHP 的时间函数获取的时间就将比当地的北京时间少 8 个小时。因此，要获取本地当前的时间，就必须更改 PHP 语言中的时区设置。更改时区设置两种方法：在 PHP.INI 文件中设置和通过 date_default_timezone_set 函数设置。

10.1.1　在 PHP.INI 文件中设置时区

在 PHP.INI 文件中设置时区需要定位到[date]下的;date.timezone =选项，去掉前面的分号，并设置它的值为当地所在时区使用的时间。

例如，如果当地所在时区为东八区，就可以设置 date.timezone =的值为：PRC（中华人民共和国）、Asia/Hong_Kong（香港）、Asia/Shanghai（上海）或 Asia/Urumqi（乌鲁木齐）等，这些都是东八区的时间，如图 10.1 所示。

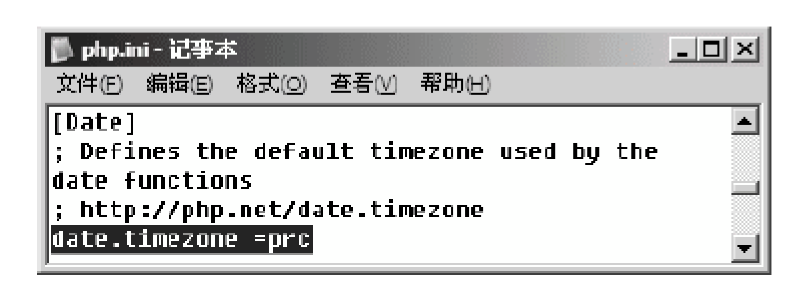


图 10.1　设置 PHP 的时区

设置完成后保存文件，并重新启动 Apache 服务器。

10.1.2　通过 date_default_timezone_set 函数设置时区

在应用程序中，在日期、时间函数之前使用 date_default_timezone_set()函数同样可以完成对时区的设置。date_default_timezone_set()函数的语法如下：

```
date_default_timezone_set(timezone);
```

参数 timezone 为 PHP 可识别的时区名称，如果 PHP 无法识别时区名称，则系统采用 UTC 时区。

例如，设置北京时间可以使用的时区包括：PRC、Asia/Chongqing（重庆），Asia/Shanghai（上海）或 Asia/Urumqi（乌鲁木齐），这几个时区名称是等效的。

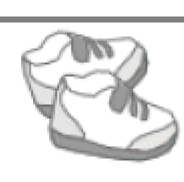
多学两招：

如果服务器使用的是零时区，则不能对 php.ini 文件直接进行修改，而只能通过 date_default_timezone_set()函数对时区进行设置。

上机演练

上机演练 1　获取不同地区的当前时间

系统的当前时间受时区限制。默认情况下，系统的当前时间是格林威治时间。用户正确取



Note

得所在的本地时间是十分重要的。本例通过 date()函数获取指定地区的当前时间，运行结果如图 10.2 所示。



图 10.2　获取指定地区的当前时间

10.2　UNIX 时间戳

视频讲解：配套资源\mr\10\video\UNIX 时间戳.exe

在日期、时间函数中，UNIX 时间戳的获取是非常重要的，有很多针对时间的操作都必须转换为时间戳之后才能够完成，例如比较时间大小、计算时间戳以及实现倒计时的功能等。在 PHP 中提供了很多获取日期、时间戳的函数，如表 10.1 所示。

表 10.1　将日期、时间转换成 UNIX 时间戳的函数

函　数	说　明
gmmktime()	获取 GMT（Greenwich Mean Time）日期的 UNIX 时间戳
microtime()	返回当前 UNIX 时间戳和微秒数
mktime()	获取一个日期的 UNIX 时间戳
strtotime()	将任何英文文本的日期时间描述解析为 UNIX 时间戳
time()	返回当前的 UNIX 时间戳

10.2.1　获取任意日期、时间的时间戳

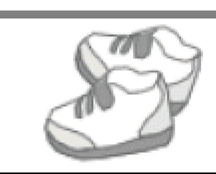
mktime()函数将一个时间转换成 UNIX 时间戳。语法如下：

```
int mktime(int hour, int minute, int second, int month, int day, int year, int [is_dst] )
```

mktime()函数根据给出的参数返回 UNIX 时间戳。其参数可以从右向左省略，任何省略的参数都会被设置成本地日期、时间的当前值（即不设置任何参数，mktime()函数获取的是本地当前日期和时间）。mktime()函数的参数说明如表 10.2 所示。

表 10.2　mktime()函数的参数说明

参　数	说　明
hour	小时数
minute	分钟数
second	秒数（一分钟之内）
month	月份数
day	天数
year	年份数，可以是两位或四位数字，0～69 对应于 2000～2069，70～100 对应于 1970～2000
is_dst	参数 is_dst 在夏令时可以被设为 1，如果不是则设为 0；如果不确定是否为夏令时则设为-1（默认值）



指点迷津：

有效的时间戳范围是格林威治时间 1901 年 12 月 13 日 20:45:54 到 2038 年 1 月 19 日 03:14:07（此范围符合 32 位有符号整数的最小值和最大值）。在 Windows 系统中此范围限制为从 1970 年 1 月 1 日到 2038 年 1 月 19 日。

例 10.1　应用 mktime()函数获取当前时间的时间戳，代码如下：（**实例位置：配套资源\mr\10\example\10.1**）

```
<?php
echo mktime();                          //当前时间戳
?>
```

运行结果为：1287628331

10.2.2　获取当前时间戳

上述讲解的 mktime()函数在不设置任何参数的情况下可以获取当前时间的时间戳，但是 PHP 也提供了专门的获取当前时间时间戳的函数，那就是 time()函数。

time()函数获取当前的 UNIX 时间戳，返回值为从 UNIX 纪元（格林威治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。语法如下：

```
int time ( void )
```

time()函数没有参数，返回值为 UNIX 时间戳的整数值。

例 10.2　应用 time()函数获取当前时间的时间戳，代码如下：（**实例位置：配套资源\mr\10\example\10.2**）

```
<?php
echo time();                            //当前时间戳
?>
```

运行结果为：1287628348

10.2.3　日期、时间转换为 UNIX 时间戳

strtotime()函数将任何英文文本的日期时间描述解析为 UNIX 时间戳。语法如下：

```
int strtotime ( string time [, int now] )
```

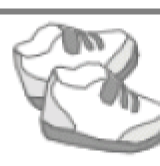
strtotime()函数接受一个包含英语日期格式的字符串并尝试将其解析为 UNIX 时间戳（自 January 1 1970 00:00:00 GMT 起的秒数），其值相对于 now 参数给出的时间，如果没有提供此参数，则使用系统当前时间。

如果参数 time 的格式是绝对时间，则 now 参数不起作用；如果参数 time 的格式是相对时间，那么其对应的时间由参数 now 来提供。

如果解析成功，则返回时间戳，否则返回 False。在 PHP5.1.0 之前本函数在失败时返回-1。

例 10.3　应用 strtotime()函数将当前时间和指定日期转换为时间戳，代码如下：（**实例位置：配套资源\mr\10\example\10.3**）

```
<?php
echo strtotime ("now"), "\n";                                   //当前时间的时间戳
echo "输出时间:".date("Y-m-d H:i:s",strtotime ("now")),"<br>";  //输出当前时间
echo strtotime ("20 October 2010"), "\n";                       //输出指定日期的时间戳
```




Note

```
echo "输出时间:".date("Y-m-d H:i:s",strtotime ("20 October 2010")),"<br>";//输出指定日期的时间
?>
```

其运行结果如图 10.3 所示。

```
1306723493 输出时间:2011-05-30 02:44:53
1287532800 输出时间:2010-10-20 00:00:00
```

图 10.3　将当前时间和指定日期转换为时间戳

指点迷津：

如果给定的年份是两位数字的格式，则其值 0～69 表示 2000～2069，70～100 表示 1970—2000。有效的时间戳通常从 Fri, 13 Dec 1901 20:45:54 GMT 到 Tue, 19 Jan 2038 03:14:07 GMT（对应于 32 位有符号整数的最小值和最大值）。并不是所有的平台都支持负的时间戳，所以日记范围就被限制为不能早于 UNIX 纪元。这意味着在 1970 年 1 月 1 日之前的日期将不能用在 Windows、一些 Linux 版本以及几个其他的操作系统中。不过 PHP 5.1.0 及更新的版本克服了此限制。

上机演练

上机演练 2　计算考试时间

计算考试时间程序是一种单纯地将两个时间点的时间戳做算术运算。本例通过 time()函数实现考试时间的计算，运行结果如图 10.4 所示。

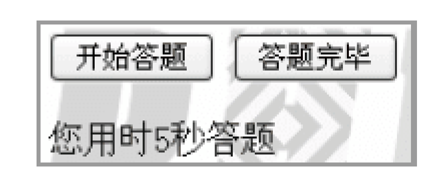


图 10.4　计算考试时间

10.3　日期和时间处理

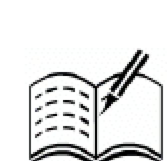
视频讲解：配套资源\mr\10\video\日期和时间处理.exe

日期和时间的处理可以分为：格式化日期和时间、获取日期和时间信息、获取本地化的日期和时间和检验日期和时间的有效性等。其使用的函数如表 10.3 所示。

表 10.3　日期和时间处理函数

函　　数	说　　明
checkdate()	验证日期的有效性
date()	格式化一个本地时间/日期
getdate()	获取日期/时间信息
gettimeofday()	获取当前时间
gmdate()	格式化一个 GMT（格林威治标准时间）/UTC 日期/时间
gmstrftime()	根据区域设置格式化 GMT/UTC 时间/日期
localtime()	获取本地时间
strftime()	根据区域设置格式化本地时间/日期





10.3.1　格式化日期和时间

date()函数对本地日期和时间进行格式化。语法如下：

```
date(string format,int timestamp)
```

参数 format 指定日期和时间输出的格式，其选项如表 10.4 所示；参数 timestamp 为可选参数，指定时间戳，如果没有指定时间戳，则使用本地时间戳 time()。

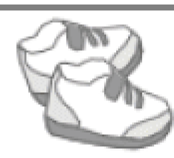


表 10.4　date()函数中参数 format 的格式选项

参　　数	说　　明
a	小写的上午和下午值，返回值为 am 或 pm
A	大写的上午和下午值，返回值为 AM 或 PM
B	Swatch Internet 标准时间，返回值为 000～999
d	月份中的第几天，有前导零的两位数字，返回值为 01～31
D	星期中的第几天，文本格式，3 个字母，返回值为 Mon～Sun
F	月份，完整的文本格式，返回值为 January～December
G	小时，12 小时格式，没有前导零，返回值为 1～12
H	小时，24 小时格式，没有前导零，返回值为 0～23
i	有前导零的分钟数，返回值为 00～59
I	判断是否为夏令时，返回值如果是夏令时则为 1，否则为 0
J	月份中的第几天，没有前导零，返回值为 1～31
l	星期数，完整的文本格式，返回值为 Sunday～Saturday
L	判断是否为闰年，返回值如果是闰年则为 1，否则为 0
m	数字表示的月份，有前导零，返回值为 01～12
M	3 个字母缩写表示的月份，返回值为 Jan～Dec
n	数字表示的月份，没有前导零，返回值为 1～12
o	与格林威治时间相差的小时数，如 0200
r	RFC 822 格式的日期，如 Thu，21 Dec 2000 16:01:07 +0200
s	秒数，有前导零，返回值 00～59
S	每月天数后面的英文后缀，两个字符，如 st、nd、rd 或 th。可以和 j 一起使用
t	指定月份所应有的天数
T	本机所在的时区
U	从 UNIX 纪元（January 1 1970 00:00:00 GMT）开始至今的秒数
w	星期中的第几天，数字表示，返回值为 0～6
W	ISO-8601 格式年份中的第几周，每周从星期一开始
y	2 位数字表示的年份，返回值如 88 或 08
Y	4 位数字完整表示的年份，返回值如 1998、2008
z	年份中的第几天，返回值为 0～366
Z	时差偏移量的秒数。UTC 西边的时区偏移量总是负的，UTC 东边的时区偏移量总是正的，返回值为-43200～43200

例 10.4　应用 date()函数设置不同的 format 值，输出不同格式的时间，代码如下：（**实例位置：配套资源\mr\10\example\10.4**）

```
<?php
echo "单个变量：".date("m月");                              //输出单个日期
```




```
echo "<p>";
echo "组合变量：".date("Y-m-d");                    //输出组合参数
echo "<p>";
echo "详细的日期及时间：".date("Y-m-d H:i:s");        //输出详细的日期和时间参数
echo "<p>";
echo "中文格式日期及时间：".date("Y年m月d日  H时i分s秒");   //输出中文格式时间
?>
```

运行结果如图 10.5 所示。

指点迷津：

在运行本章的程序时，也许有些读者得到的时间和系统时间并不相同，这不是程序的问题。因为在 PHP 语言中默认设置的是标准的格林威治时间，而不是北京时间。如果出现了时间不符的情况，读者可参考 10.1 节的相关内容。

例 10.5 应用 date()和 time()函数获取系统当前时间和时间戳，具体代码如下：（**实例位置：配套资源\mr\10\example\10.5**）

```
<?php
    echo date("Y年m月d日  H时i分s秒");        //获取当前时间
    echo "<br>";                              //换行
    echo time();                              //获取当前时间戳
?>
```

运行效果如图 10.6 所示。

单个变量：05月

组合变量：2011-05-30

详细的日期及时间：2011-05-30 02:45:20

中文格式日期及时间：2011年05月30日 02时45分20秒

图 10.5 date()函数输出不同格式的当前时间

2011年05月30日 02时47分24秒
1306723644

图 10.6 获取当前时间的时间戳

10.3.2 获取日期和时间信息

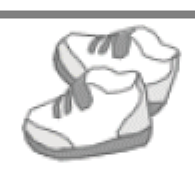
getdate()函数获取日期和时间指定部分的相关信息。语法如下：

```
array getdate(int timestamp)
```

getdate()函数返回数组形式的日期、时间信息，如果没有时间戳，则以当前时间为准。getdate()函数返回的关联数组元素的说明如表 10.5 所示。

表 10.5 getdate()函数返回的关联数组元素的说明

键　名	说　明	返　回　值
seconds	秒	返回值为 0～59
minutes	分钟	返回值为 0～59
hours	小时	返回值为 0～23
mday	月份中第几天	返回值为 1～31
wday	星期中第几天	返回值为 0（表示星期日）～6（表示星期六）
mon	月份数字	返回值为 1～12





续表

键　　名	说　　明	返　回　值
year	4 位数字表示的完整年份	返回值如 2010 或 2011
yday	一年中第几天	返回值为 0～365
weekday	星期几的完整文本表示	返回值为 Sunday～Saturday
month	月份的完整文本表示	返回值为 January～December
0	自从 UNIX 纪元开始至今的秒数，和 time()的返回值以及用于 date()的值类似	系统相关，典型值为从-2147483648 到 2147483647

getdate()函数比较适合获取当前日期是一年、月份或星期中的第几天。虽然它也可以获取当前的日期，但是由于其获取的为返回值数组，所以更适合获取时间中某个特定的值。

例 10.6　通过 getdate()函数获取当前日期以及当前日期是一年中的第几天和当月的第几天，具体代码如下：**（实例位置：配套资源\mr\10\example\10.6）**

```
<?php
$arr = getdate();                                    //使用getdate()函数将当前信息保存
$month=date("m月");                                  //输出单个日期
$year=date("Y年");
echo "当前日期：".$arr[year]."-".$arr[mon]."-".$arr[mday]." ";  //返回当前的日期信息
echo "<P>";
echo "今天是".$year."中的第".$arr[yday]."天";        //输出今天是一年中的第几天
echo "<p>";
echo "今天是".$year.$month."的第".$arr[mday]."天";   //输出今天是本月中的第几天
?>
```

运行结果如图 10.7 所示。

当前日期：2011-5-30

今天是2011年中的第149天

今天是2011年05月的第30天

图 10.7　getdate()函数获取当前时间信息

10.3.3　检验日期和时间的有效性

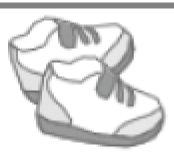
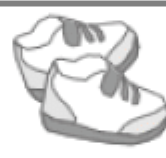
一年 12 个月、一个月 31 天（或 30 天，2 月为 28 天，闰年为 29 天），一星期 7 天……这些都是基本常识。但计算机并不能自己分辨数据的对与错，只是依靠开发者提供的功能去执行或检查。在 PHP 中通过 checkdate()函数检验日期和时间的有效性。语法如下：

```
bool checkdate(int month,int day,int year)
```

参数 month 的有效值为 1～12；参数 day 的有效值为当月的最大天数，例如，1 月为 31 天，2 月为 29 天（闰年）；参数 Year 的有效值为 1～32767。如果验证的日期有效，则返回 True，否则返回 False。

例 10.7　验证 2010 年 2 月份到底是 28 天还是 29 天，具体代码如下：**（实例位置：配套资源\mr\10\example\10.7）**

```
<?php
    $year = 2010;                                    //2010年
```

```
        $month = 2;                                    //2月份
        $one_day = 28;                                 //28天
        if(checkdate($month,$one_day,$year)){
            echo "2010年2月份是28天";
        }else{
            echo "2010年2月份是29天";
        }
    ?>
```

运行结果为：2010 年 2 月份是 28 天

上机演练

上机演练 3　输出中文格式的日期和时间

输出中文格式的时间其实在前面的操作中已经提到过，但由于该操作具有的重要性，此处仍对其进行介绍。本例采用 date()函数实现中文格式的日期和时间的输出，运行效果如图 10.8 所示。

上机演练 4　检验日期和时间的有效性

在 PHP 中通过 checkdate()函数检验日期和时间的有效性。运行效果如图 10.9 所示。

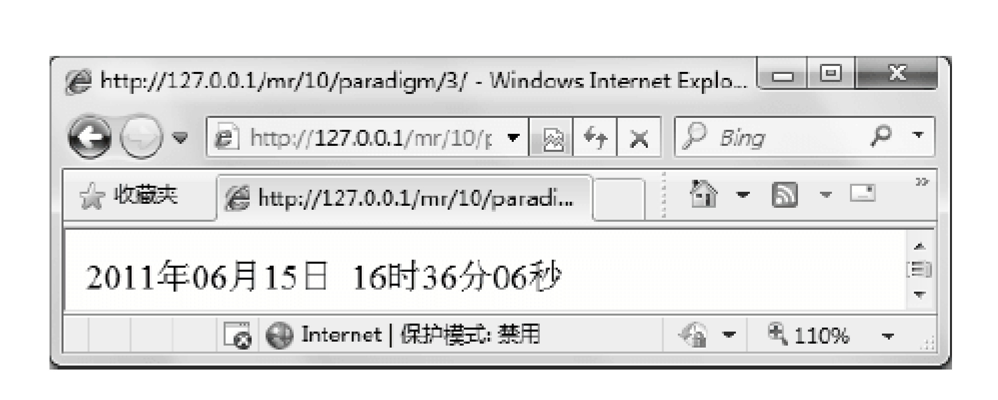


图 10.8　输出中文格式的日期和时间

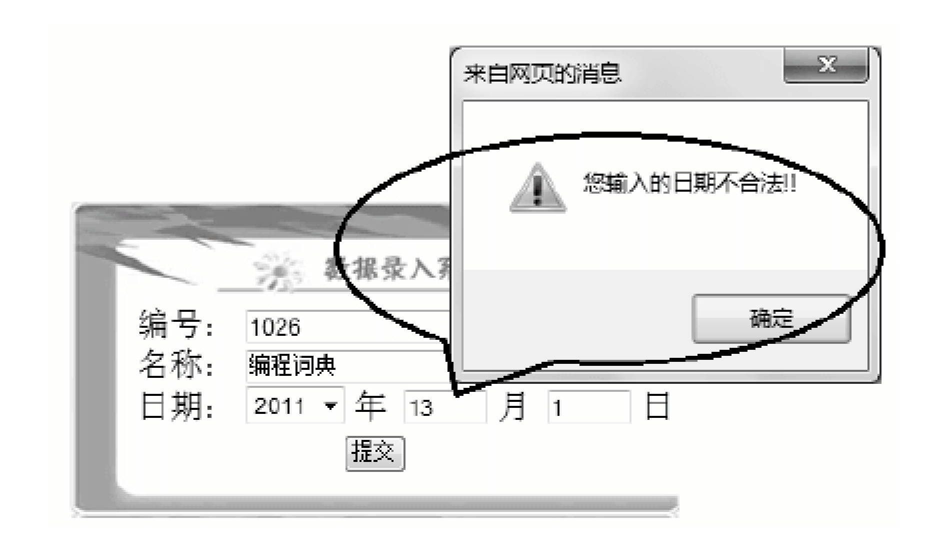


图 10.9　检验日期和时间的有效性

本 章 摘 要

1．通过 php.ini 文件为服务器设置正确的时间。

2．获取指定时间的时间戳，或者获取系统当前时间的时间戳，完成时间、时间戳格式的转换。

3．获取系统当前格式化的时间，验证日期时间的有效性。

4．应用日期时间函数完成倒计时、网页闹钟、比较时间大小和计算考试剩余时间等程序。

习　　题

1．下面哪个函数可以实现当地时间的设置？（　　）

A．date　　　B．time　　　C．date_default_timezone_set

2．获取当前时间戳需要使用的函数是（　　）。





A．date　　B．time　　C．date_default_timezone_set　　D．substr

3．检验日期和时间的有效性需要使用的函数是（　　）。

A．date　　B．time　　C．checkdate

4．获取时间戳的微秒数需要使用的函数是（　　）。

A．time　　B．date　　C．microtime

5．看下面一行代码：

```
echo date("Y-m-d")
```

程序运行后，输出日期的格式为（　　）。

A．X 年 X 月 X 日　　B．X-X-X　　C．X-X–X　　X: X:X

6．获取 2010 年 1 月 1 日的时间戳代码为（　　　　　　　　）。

7．将系统时间设置为当前本地时间的代码为（　　　　　　　　）。

8．利用 mktime 函数获取当前时间的代码为（　　　　　　　　）。

9．计算 2010 年 12 月 10 日是星期几的代码为（　　　　　　　　）。

10．程序运行后，下面代码显示的内容为（　　　　　　　　）。

```
<?php
    $time1 = strtotime(date("Y-m-d"));
    $time2 = strtotime("2008-09-18");
    if($time1==$time2){
        echo "<script>alert('勿忘国耻！');window.location.href='index.php';</script>";//给出提示信息
    }else{
        echo "今天不是一个特殊的日子！";
    }
?>
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 9 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　倒计时

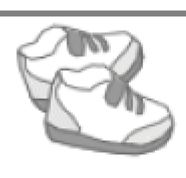
倒计时是大家在生活中经常会用到的一个功能，例如，2010 年上海世博会的倒计时，2011 年春节的倒计时等。下面应用 PHP 的日期、时间函数为 2011 年元旦设计一个倒计时程序，运行效果如图 10.10 所示。

距离2011年元旦还有72 天！！！

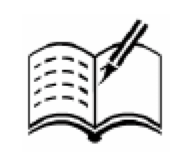
图 10.10　倒计时

实战模拟 2　比较两个时间的大小

在 PHP 语言中要完成对两个时间大小的比较，必须先将时间转换为时间戳，然后才可以进行比较。而将时间转换为时间戳可以通过 strtotime()函数完成。其运行效果如图 10.11 所示。







时间：2010年10月10日 10时10分10秒

早于 2011-11-11 11:11:11

统计用户的在线时间

TONG JI YONG HU DE ZAI XIAN SHI JIAN



图 10.11　比较两个时间戳的大小

实战模拟 3　网页闹钟

闹钟是人们生活中经常使用的一个小工具，它会在指定时间叫你起床。在 Web 程序中也可以应用这个原理开发出一个“网页闹钟”程序，以提示用户在指定的时间或时间段内要做什么工作。当系统的当前时间运行到 9 月 18 日，给出提示信息“勿忘国耻！”。其运行效果如图 10.12 所示。



图 10.12　网页闹钟

实战模拟 4　计算程序的运行时间

在百度中，当大家执行一个查询操作时，在获取到查询结果后，页面中就会出现一行文字，提示您根据关键字搜索到多少个结果以及搜索所用的时间。这个时间就是程序在执行这个搜索时所用的时间，那么它是如何实现的呢？下面模仿它来做一个计算程序运行时间的小程序。其运行结果如图 10.13 所示。

距离2011年元旦还有72 天!!!

程序的运行时间为： 0.000304937362671 秒

图 10.13　计算程序运行时间



图形图像处理

（ 自学视频、源程序：配套资源\mr\11\）

由于有GD库的强大支持，图像处理功能可以说是PHP的强项，其便捷易用、功能强大。另外，PHP图形化类库——Jpgraph也是一款非常好用和强大的图形处理工具，可以绘制各种统计图和曲线图，也可以自定义设置颜色和字体等元素。

图像处理技术中的经典应用是绘制饼形图、柱形图和折线图，这是对数据进行图形化分析的最佳方法。本章将分别对GD2函数及Jpgraph类库进行详细讲解。

学习摘要：

- 了解GD2函数库
- 设置GD2函数库
- 常用图像处理技术
- 运用Jpgraph类库绘制图像






11.1 了解 GD2 函数库

目前，在 Web 开发领域 PHP 已经被广泛应用，互联网上已经有近半数的站点采用 PHP 作为核心语言。PHP 不仅可以生成 HTML 页面，而且可以创建和操作二进制形式数据，如图像和文件等，其中，使用 PHP 操作图形可以通过 GD2 函数库来实现。使用 GD2 函数库可以在页面中绘制各种图形图像及统计图，如果与 Ajax 技术相结合，还可以制作出各种强大的动态图表。

GD2 函数库是一个开放的、动态创建图像的、源代码公开的函数库，读者可以从其官方网站 http://www.boutell.com/gd 上下载最新版本的 GD2 库。目前，GD2 库支持 GIF、PNG、JPEG、WBMP 和 XBM 等多种图像格式。

11.2 设置 GD2 函数库

在 PHP5 中，GD2 函数库已经作为扩展被默认安装，但目前有些版本还需要对 php.ini 文件进行设置来激活 GD2 函数库。激活过程为：使用文本编辑工具（如记事本等）打开 php.ini 文件，将该文件中的;extension=php_gd2.dll 选项前的分号“;”删除，然后保存修改后的文件，并重新启动 Apache 服务器，即可激活 GD2 函数库，如图 11.1 所示。

在成功激活 GD2 函数库后，可以通过 phpinfo()函数来获取 GD2 函数库的安装信息，验证 GD 库是否安装成功。在 Apache 的默认站点目录中编写 phpinfo.php 文件，并在该文件中编写如下代码：

```
<?php
    phpinfo();        //输出PHP配置信息
?>
```

在 IE 浏览器的地址栏中输入 http://127.0.0.1/phpinfo.php，按 Enter 键后，如果在打开的页面中检索到如图 11.2 所示的 GD2 函数库的安装信息，则说明 GD 库安装成功，这样开发人员就可以在程序中使用 GD2 函数库创建图形图像了。



图 11.1 激活 GD2 函数库

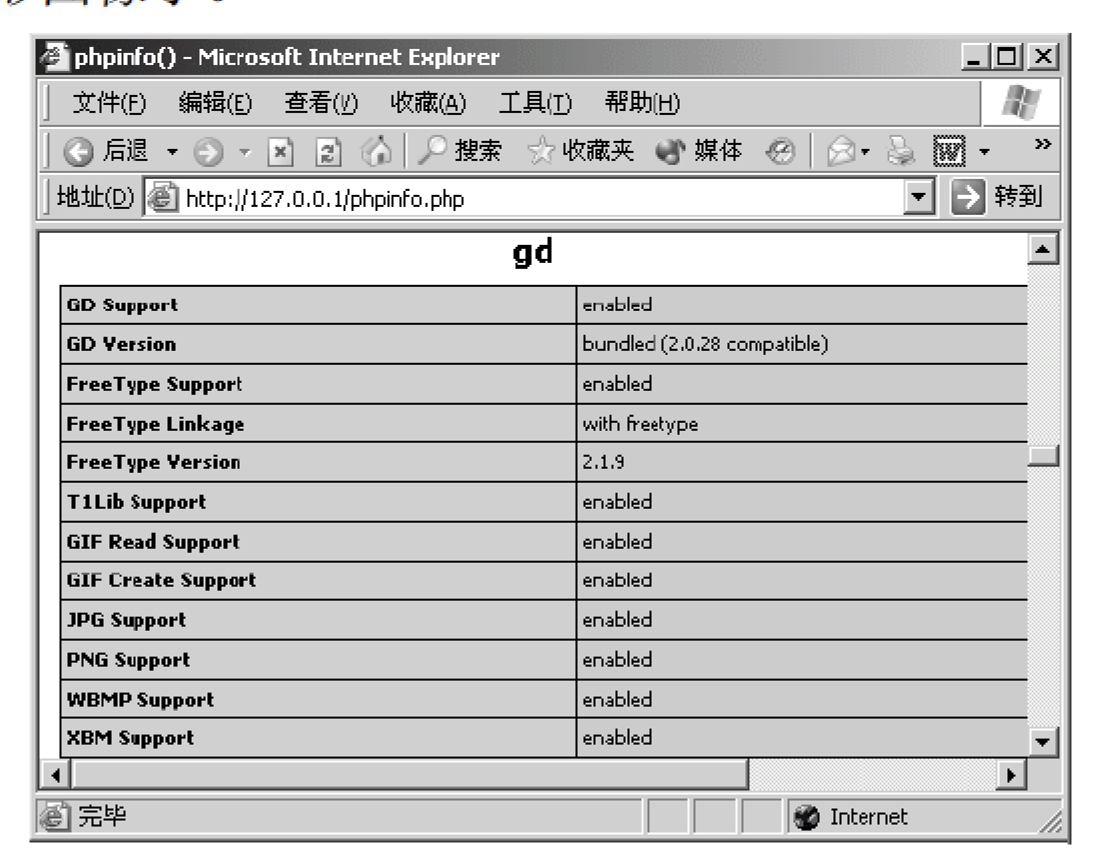


图 11.2 GD2 函数库的安装信息





11.3 常用图像处理技术

视频讲解：配套资源\mr\11\video\常用图像处理技术.exe

PHP 中的 GD2 函数库用于创建或处理图像，通过它可以生成统计图表、动态图形、缩略图和图形验证码等。在 PHP 中，对图像的操作可以分为以下 4 个步骤：

（1）创建画布。

（2）在画布上绘制图形。

（3）保存并输出结果图像。

（4）销毁图像资源。

11.3.1 创建画布

GD2 函数库在图像图形绘制方面的功能非常强大，开发人员既可以在已有图片的基础上进行绘制，也可以在没有任何素材的基础上绘制，在这种情况下，首先要创建画布，之后所有操作都将依据所创建的画布进行。在 GD2 函数库中创建画布应用 imagecreate()函数，其语法如下：

```
resource imagecreate ( int x_size, int y_size )
```

该函数用于返回一个图像标识符，参数 x_size、y_size 为图像的尺寸，单位为像素（pixel）。

例 11.1 通过 imagecreate()函数创建一个宽 400 像素、高 100 像素的画布，并且设置画布背景颜色的 RGB 值为 200、60、60，然后输出一个 JPEG 格式的图像，具体代码如下：（**实例位置：配套资源\mr\11\example\11.1**）

```
<?php
    header("Content-type:text/html;charset=utf-8");          //设置页面的编码风格
    header("Content-type:image/jpg");                         //告知浏览器输出的是图片
    $image=imagecreate(400,100);                              //设置画布的大小
    $bgcolor=imagecolorallocate($image,200,60,60);            //设置画布的背景颜色
    imagejpeg($image);                                        //输出图像
    imagedestroy($image);                                     //销毁图像
?>
```

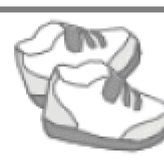
在上面的代码中，应用 imagecreate()函数创建一个基于普通调色板的画布，通常支持 256 色。其中通过 imagecolorallocate()函数设置画布的背景颜色；通过 imagegjpeg()函数输出图像；通过 imagedestroy()函数销毁图像资源。其运行效果如图 11.3 所示。



图 11.3 创建画布

11.3.2 颜色处理

应用 GD2 函数绘制图形需要为图形中的背景、边框和文字等元素指定颜色，在 GD2 中使



用 imagecolorallocate()函数设置颜色，其语法如下：

```
int imagecolorallocate ( resource image, int red, int green, int blue)
```

其中参数 image 是 imagecreatetruecolor()函数的返回值；red、green 和 blue 分别是所需要的颜色的红、绿、蓝成分，这些参数是 0～255 的整数或者十六进制的 0x00～0xFF。

imagecolorallocate()函数返回一个标识符，代表由给定的 RGB 成分组成的颜色。



提示：

如果是第一次调用 imagecolorallcate()函数，那么它将完成背景颜色的填充。

例 11.2 通过 imagecreate()函数创建一个宽 685 像素、高 180 像素的画布，通过 imagecolorallocate()函数为画布设置背景颜色及图像的颜色，并且输出创建的图像。（**实例位置：配套资源\mr\11\example\11.2**）

首先，通过 imagecreate()函数创建一个宽 685 像素、高 180 像素的画布。然后，通过 imagecolorallocate()函数为画布设置背景颜色以及图像的颜色。接着，通过 imageline()函数绘制一条白色的直线。最后，完成图像的输出和资源的销毁。其代码如下：

```
<?php
    header("Content-Type:text/html;charset=utf-8");            //设置页面的编码风格
    header("Content-Type:image/jpeg");                         //告知浏览器输出的是一个图片
    $image=imagecreate(685,180);                               //设置画片大小
    $bgcolor=imagecolorallocate($image,200,60,120);            //设置图片的背景颜色
    $write=imagecolorallocate($image,200,200,250);             //设置线条的颜色
    imageline($image,20,20,650,160,$write);                    //画一条线
    imagejpeg($image);                                         //输出图像
    imagedestroy($image);                                      //销毁图片
?>
```

运行结果如图 11.4 所示。

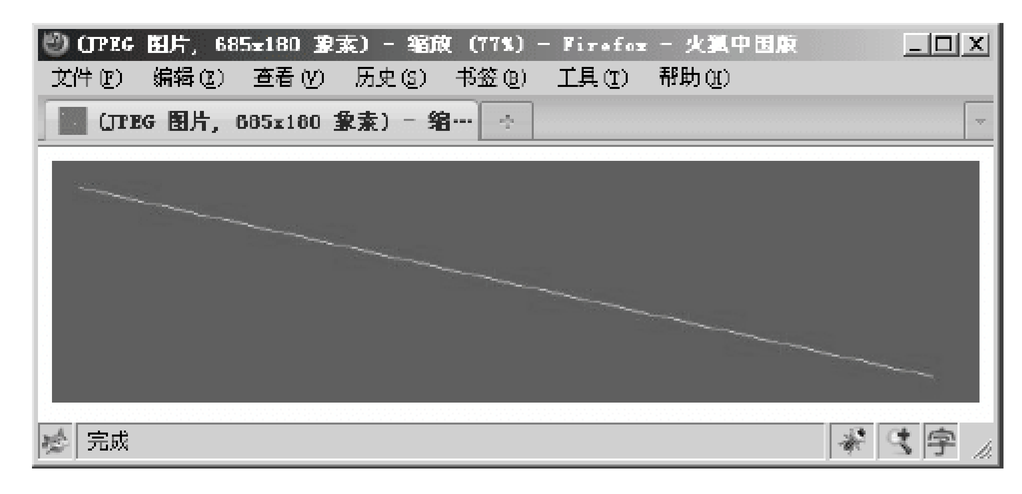


图 11.4　绘制一条白色直线

11.3.3　绘制文字

在 PHP 中的 GD 库既可以绘制英文字符串，也可以绘制中文汉字。绘制英文字符串应用 imagestring()函数。其语法如下：

```
bool imagestring ( resource image, int font, int x, int y, string s, int col )
```

imagestring()函数用 col 颜色将字符串 s 绘制到 image 所代表的图像的（x，y）坐标处（这是字符串左上角坐标，整幅图像的左上角为（0，0））。如果 font 是 1、2、3、4 或 5，则使用内置字体。

绘制中文汉字应用 imagettftext()函数，其语法如下：

```
array imagettftext (resource image, float size, float angle, int x, int y, int color, string fontfile, string text)
```



imagettftext()函数的参数说明如表 11.1 所示。

表 11.1 imagettftext()函数的参数说明

参 数	说 明
image	图像资源
size	字体大小。根据 GD 版本不同，应以像素大小（GD1）或点大小（GD2）指定
angle	字体的角度，顺时针计算，0° 为水平，也就是 3 点钟的方向（由左到右），90° 则为由下到上的文字
x	文字的 x 坐标值。它设定了第一个字符的基本点
y	文字的 y 坐标值。它设定了字体基线的位置，不是字符的最底端
color	文字的颜色
fontfile	字体的文件名称，也可以是远端的文件
text	字符串内容

指点迷津：

在 GD2 函数库中支持的是 UTF-8 编码格式的中文，所以在通过 imagettftext()函数输出中文字串时，必须保证中文字符串的编码格式是 UTF-8，否则中文将不能正确输出。如果定义的中文字符串是 GB2312 简体中文编码，那么要通过 iconv()函数对中文字符串的编码格式进行转换。

例 11.3 通过 imagestring()函数水平地绘制一行字符串“I like PHP”。**（实例位置：配套资源\mr\11\example\11.3）**

首先，创建一个画布。然后，定义画布背景颜色和输出字符串的颜色。接着，通过 imagestring()函数水平地绘制一行英文字符串。最后，输出图像并且销毁图像资源。其代码如下：

```
<?php
 header("Content-Type:text/html;charset=utf-8");          //设置页面的编码风格
 header("Content-Type:image/jpeg");                        //告知浏览器输出的是一张图片
 $image=imagecreate(300,80);                               //创建画布的大小
 $bgcolor=imagecolorallocate($image,200,60,90);            //设置背景颜色
 $write=imagecolorallocate($image,0,0,0);                  //设置文字颜色
 imagestring($image,5,80,30,"I Like PHP","$write");        //书写英文字符
 imagejpeg($image);                                        //输出图像
 imagedestroy($image);                                     //销毁图像
?>
```

运行结果如图 11.5 所示。

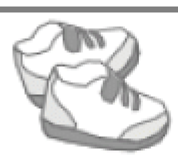
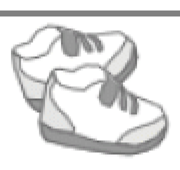


图 11.5 绘制英文字符串

例 11.4 通过 imagettftext()函数水平地绘制一行中文字符串。**（实例位置：配套资源\mr\11\ example\11.4）**

首先，创建一个画布，并定义画布背景颜色和输出字符串的颜色。然后，定义中文字符串使用的字体以及要输出的中文字符串的内容。接着，通过 imagettftext()函数水平地绘制一行中文字符串。最后，输出图像并且销毁图像资源。其代码如下：

```
<?php
    header("Content-Type:text/html;charset=utf-8");              //设置文件编码格式
```

```
    header("Content-Type:image/jpeg");                          //告知浏览器所要输出图像的类型
    $image=imagecreate(800,150);                                //创建画布
    $bgcolor=imagecolorallocate($image,0,200,200);              //设置图像背景色
    $fontcolor=imagecolorallocate($image,200,80,80);            //设置字体颜色为黑色
    $font="FZSHHJW.TTF";                                        //定义字体
    $string="明日科技";                                          //定义输出中文
    imagettftext($image,80,5,100,130,$fontcolor,$font,$string); //写TTF文字到图中
    imagejpeg($image);                                          //建立JPEG图形
    imagedestroy($image);                                       //释放内存空间
?>
```

运行结果如图 11.6 所示。

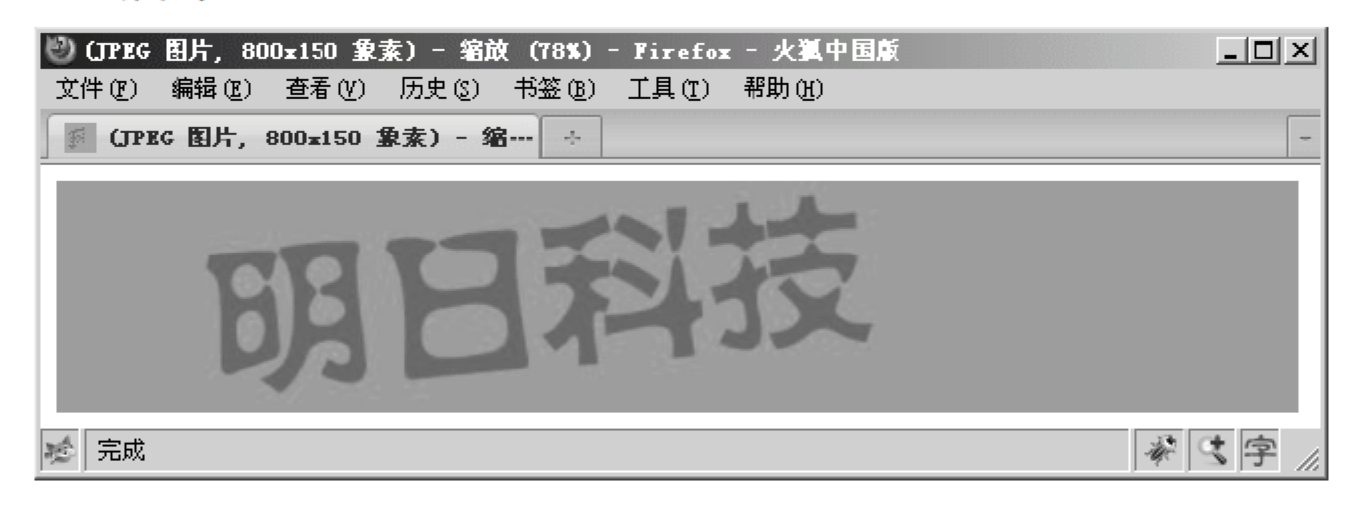


图 11.6　绘制中文字符串

多学两招：

由于 imagettftext()函数只支持 UTF-8 编码，所以如果创建的网页的编码格式使用 GB2312，那么在应用 imagettftext()函数输出中文字符串时，必须应用 iconv()函数将字符串的编码格式由 GB2312 转换为 UTF-8，否则在输出时将会出现乱码。在例 11.4 中之所以没有进行编码格式转换，是因为创建的文件默认使用的是 UTF-8 编码。

例 11.5　在用户注册功能模块中，为了提高站点的安全性，避免由于网速慢造成用户注册信息的重复提交，往往会在用户注册表单中增加验证码功能。下面应用 GD2 函数库中的函数生成一个数字验证码，其具体步骤如下。（**实例位置：配套资源\mr\11\example\11.5**）

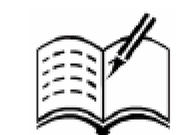
（1）设置页面的编码风格为 UTF-8。

（2）告知浏览器输出的是一个 JPEG 格式的图像。

（3）利用 GD2 函数将 rand 函数生成的验证码以图像的形式输出。其代码如下：

```
<?php
    header("Content-Type:text/html;charset=utf-8");        //设置编码风格
    header("Content-Type:image/jpeg");                      //设置图片格式
    $image=imagecreate(250,100);                            //创建画布
    $bgcolor=imagecolorallocate($image,250,180,180);        //设置背景颜色
    $fontcolor=imagecolorallocate($image,30,30,30);         //设置字体颜色
    $font="STCAIYUN.TTF";                                   //设置字体
    for($a=0;$a<4;$a++){                                    //循环语句
        $rand.=dechex(rand(0,15));
    }
    $string=$rand;
    imagefttext($image,50,7,40,80,$fontcolor,$font,$string);   //输出验证码
```





```
        imagejpeg($image);
        imagedestroy($image);
    ?>
```

运行结果如图 11.7 所示。



图 11.7　生成图像验证码

11.3.4　输出图像

PHP 作为一种 Web 语言，无论是解析出的 HTML 代码还是二进制的图片最终都要通过浏览器显示。应用 GD2 函数绘制的图像首先需要用 header()函数发送 HTTP 头信息给浏览器，告知所要输出图像的类型，然后应用 GD2 函数库中的函数完成图像输出。

header()函数用于向浏览器发送 HTTP 头信息，其语法如下：

```
void header ( string string [, bool replace [, int http_response_code]] )
```

参数 string：发送的标头；参数 replace：如果一次发送多个标头，对于相似的标头是替换还是添加，如果是 False，则强制发送多个同类型的标头，默认是 True，即替换；参数 http_response_code：强制 HTTP 响应为指定值。

header()函数可以实现如下 4 种功能。

（1）重定向，这是最常用的功能，如：

```
header("Location: http://www.mrbccd.com");
```

（2）强制客户端每次访问页面时获取最新资料，而不是使用存在于客户端的缓存，如：

```
//设置页面的过期时间（用格林威治时间表示）
header("Expires:     Mon,     08     Jul     2008     08:08:08     GMT");
//设置页面的最后更新日期（用格林威治时间表示），使浏览器获取最新资料
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . "GMT");
header("Cache-Control: no-cache,     must-revalidate");          //控制页面不使用缓存
header("Pragma: no-cache");          //参数（与以前的服务器兼容），即兼容HTTP1.0协议
header("Content-type:     application/file");                   //输出MIME类型
header("Content-Length:     227685");                           //文件长度
header("Accept-Ranges:     bytes");                             //接受的范围单位
//默认时文件保存对话框中的文件名称
header("Content－Disposition: attachment; filename=$filename");     //实现下载
```

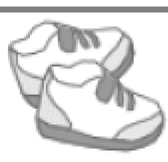
（3）输出状态值到浏览器，控制访问权限，如：

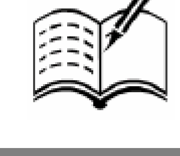
```
header('HTTP/1.1 401 Unauthorized');
header('status: 401 Unauthorized');
```

（4）完成文件的下载，如：

```
header("Content-type: application/x-gzip");
header("Content-Disposition: attachment; filename=文件名");
header("Content-Description: PHP3 Generated Data"); >
```

在应用的过程中，唯一需要改动的就是 filename，即将 filename 替换为要下载的文件。



Note

imagegif()函数以 GIF 格式将图像输出到浏览器或文件，其语法如下：

```
bool imagegif ( resource image [, string filename] )
```

参数 image：是 imagecreate()或 imagecreatefromgif()等创建图像函数的返回值，图像格式为 GIF，如果应用 imagecolortransparent()函数，则使图像设置为透明，图像格式为 GIF；参数 filename：可选参数，如果省略，则原始图像流将被直接输出。

imagejpeg()和 imagepng()函数的使用方法与 imagegif()函数类似，这里不再赘述。至于图像输出函数的应用，在前面的 4 个实例中都已经使用过，这里不再重新举例。

11.3.5 销毁图像

在 GD2 函数库中，通过 imagedestroy()函数来销毁图像，释放内存。其语法如下：

```
bool imagedestroy ( resource image )
```

imagedestroy()释放与 image 关联的内存。image 是由图像创建函数返回的图像标识符，如 imagecreatetruecolor()。

有关销毁图像函数的应用在前面的 4 个实例中都已经使用过，这里不再重新举例。

上机演练

上机演练 1　应用 GD2 函数填充几何图形

使用 GD2 函数不仅可以绘制线条图形，而且可以绘制填充图形，如填充圆形、填充矩形等。本例通过 GD2 函数填充圆形和矩形，其运行效果如图 11.8 所示。

上机演练 2　应用 GD2 函数在照片上添加文字

PHP 中的 GD 库支持中文，但必须要以 UTF-8 格式的参数来进行传递，如果使用 imageString()函数直接绘制中文字符串则会出现乱码，这是因为 GD2 对中文只能接收 UTF-8 编码格式，并且默认使用英文字体；所以要输出中文字符串，则必须对中文字符串进行转码，并设置中文字符使用的字体；否则，输出的只能是乱码。本例实现 GD2 函数输出中文字符串，并且通过 GD2 函数将中文字符串在照片上输出，其运行结果如图 11.9 所示。



图 11.8　填充几何图形

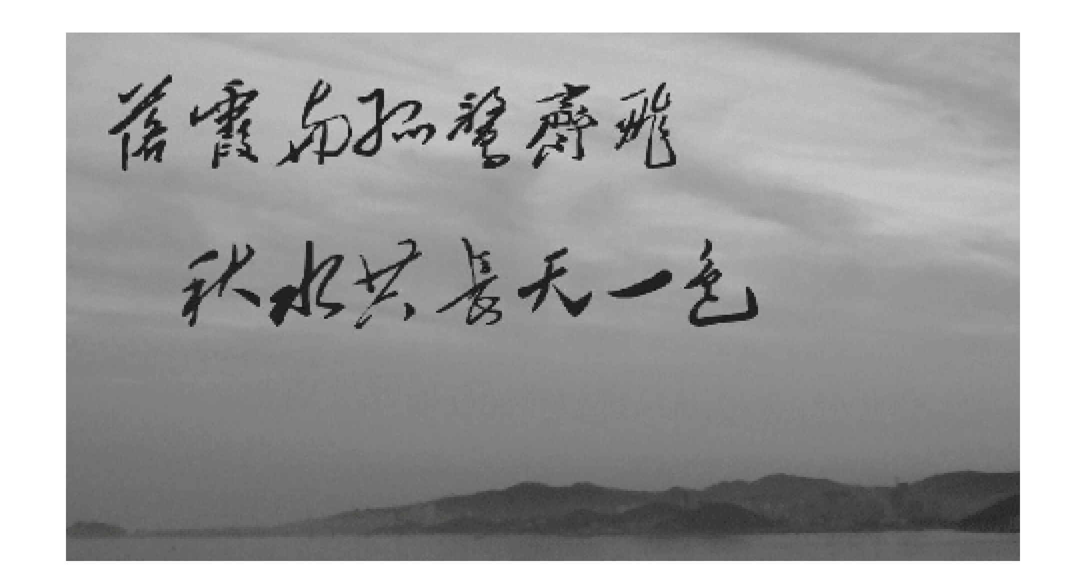


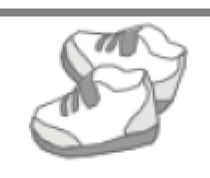
图 11.9　在照片上输出中文字符串

上机演练 3　应用 GD2 函数为图片添加图像水印

应用 GD2 函数为上传图片添加一个图像水印，其运行结果如图 11.10 所示。

上机演练 4　应用 GD2 函数生成图形验证码

验证码技术的应用是为了提高站点的安全性，避免因网页运行速度慢而造成数据的重复提交。本例通过 JavaScript 脚本和 GD2 函数开发一个无刷新验证码，其运行结果如图 11.11



所示。



图 11.10　为图片添加图像水印



图 11.11　应用 GD2 函数生成图形验证码

11.4　运用 Jpgraph 类库绘制图像

视频讲解：配套资源\mr\11\video\运用 Jpgraph 类库绘制图像.exe

Jpgraph 是基于 GD2 函数库编写的主要用于创建统计图的类库，其在绘制统计图方面不仅功能非常强大，而且代码编写方便，使用它只需简单的几行代码就可以绘制出非常复杂的统计图效果，从而很大程度地提高编程人员的开发效率。

11.4.1　Jpgraph 类库简介

Jpgraph 类库是一个可以应用在 PHP4.3.1 以上版本的用于图形图像绘制的类库。该类库完全基于 GD2 函数库编写，并提供了多种方法创建各类统计图，包括坐标图、柱形图、饼形图等。使用 Jpgraph 类库可使复杂的统计图编写工作变得简单，大大提高开发者的开发效率，在现今的 PHP 项目中被广泛应用。

多学两招：

要运用 Jpgraph 类库，首先必须了解它如何下载，都有哪些版本以及都适用于哪些环境。Jpgraph 类库包括 Jpgraph 1.x 系列、Jpgraph 2.x 系列和 Jpgraph 3.x 系列。

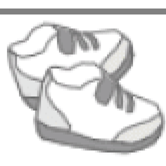
（1）Jpgraph 1.x 系列仅适用于 PHP4 环境，在 PHP5 下不能工作。

（2）Jpgraph 2.x 系列仅适用于 PHP5 环境（>= 5.1.x），在 PHP4 环境下不能工作。目前 2.x 系列的最新版本是 2.3.4。

（3）JpGraph 3.x 系列是 2.x 的升级版本。目前最新的版本是 3.0.6。

11.4.2　Jpgraph 类库的安装

在安装 Jpgraph 前，首先需要下载该类库的压缩包，Jpgraph 类库的压缩包主要有两种形式：ZIP 格式和 TAR 格式。如果是 Linux/UNIX 平台，可以选择 TAR 格式的压缩包，如果是微软的





WIN32 平台，则选择上述两种格式的任一种都可以。Jpgraph 类库可以从其官方网站 http://www.aditus.nu/jpgraph/中下载，目前最新的版本是 2.3.4。

如果已经下载 Jpgraph 的安装包，解压后将会呈现如图 11.12 所示的目录结构。

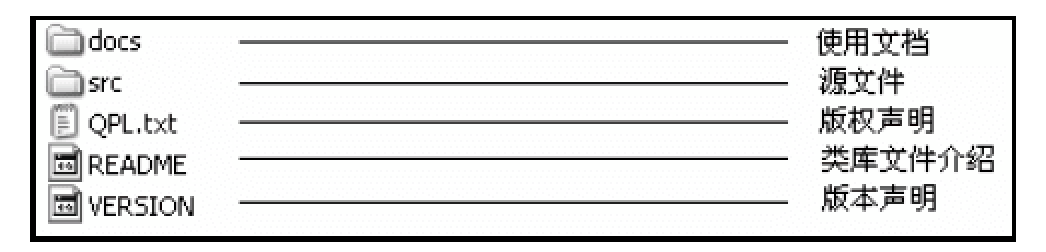


图 11.12　Jpgraph 压缩包解压后的目录结构

（1）如果希望服务器中的所有站点均有效，可以按如下步骤进行配置。

首先，解压下载的压缩包，复制图 11.12 中的 src 文件夹，并将该文件夹保存到服务器磁盘（如 c:\jpgraph）中。

然后，编辑 php.ini 文件，修改 include_path 配置项，在该项后增加 Jpgraph 类库的保存目录，如 include_path = ".;c:\jpgraph"。

最后，重新启动 Apache 服务，配置生效。

（2）如果只希望在本站点使用 Jpgraph，则只需直接将 src 文件夹复制到工程目录下即可。

通过上述两种方式都可以完成 Jpgraph 类库的安装，此时在程序中通过 require_once 语句即可完成 Jpgraph 类库的载入操作。代码如下：

```
require_once 'src/jpgraph.php';
```

指点迷津：

因为 Jpgraph 类库属于第三方的内容，所以在本书配套资源中没有提供。在运行本章中所有涉及到 Jpgraph 类库的程序时，都需要读者自己下载 Jpgraph 类库，然后复制 src 文件夹到实例根目录的上级文件夹下。

11.4.3　柱形图分析产品月销售量

例 11.6　通过 Jpgraph 类库创建柱形图，完成对产品月销售量的统计分析，具体步骤如下。**（实例位置：配套资源\mr\11\example\11.6）**

（1）将 Jpgraph 类库导入到程序中。

使用 Jpgraph 类库，首先从网站 http://www.aditus.nu/jpgraph/中下载该类库的压缩包，然后进行解压，将呈现如图 11.13 所示的目录结构。

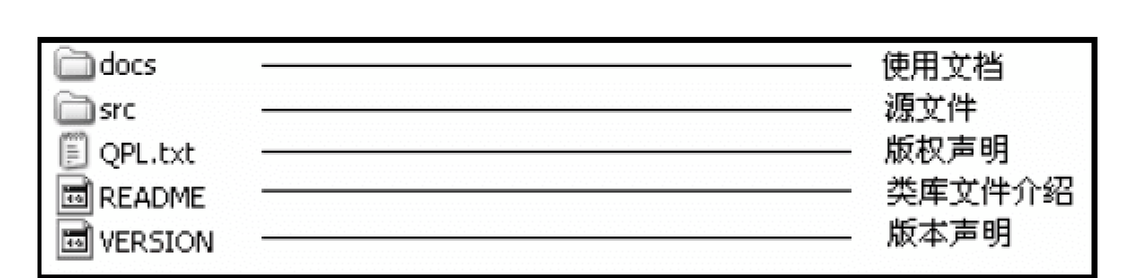


图 11.13　Jpgraph 压缩包解压后的目录结构

（2）将 src 文件夹复制到实例的根目录下。

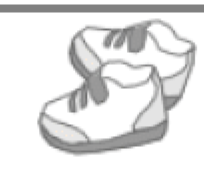
（3）创建 index.php 文件，将 Jpgraph 导入到项目中。具体操作步骤如下：

① 使用 Graph 类创建统计图对象。

② 调用 Graph 类的 SetScale 方法设置统计图的刻度样式。

③ 调用 Graph 类的 SetShadow 方法设置统计图阴影。

④ 调用 Graph 类的 img 属性的 SetMargin 方法设置统计图的边界范围。

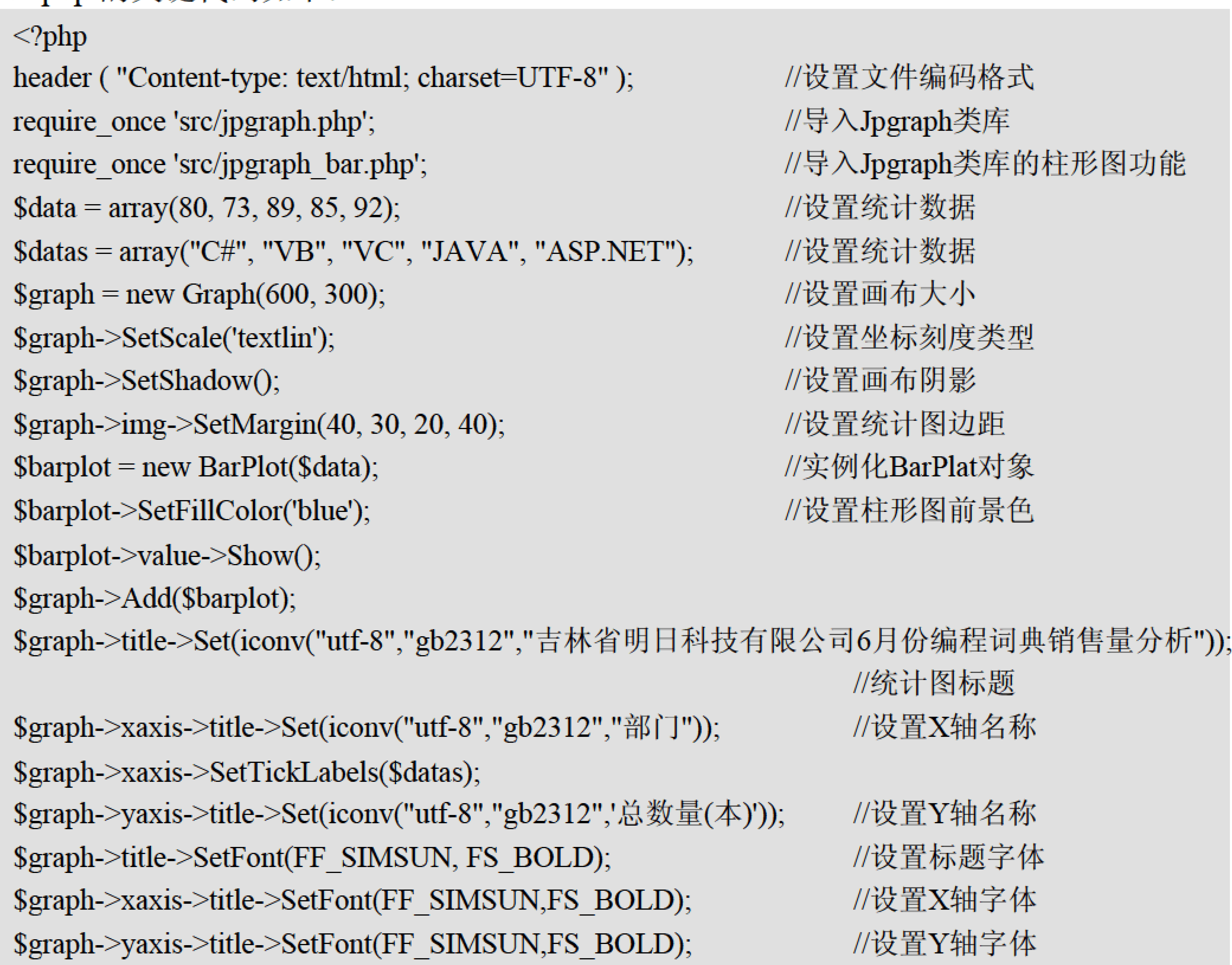




⑤ 调用 BarPlot 类创建统计图的柱状效果。

⑥ 调用 BarPlot 类的 SetFillColor 方法设置柱形图的前景色。

index.php 的关键代码如下：



```
<?php
header ( "Content-type: text/html; charset=UTF-8" );          //设置文件编码格式
require_once 'src/jpgraph.php';                                 //导入Jpgraph类库
require_once 'src/jpgraph_bar.php';                             //导入Jpgraph类库的柱形图功能
$data = array(80, 73, 89, 85, 92);                              //设置统计数据
$datas = array("C#", "VB", "VC", "JAVA", "ASP.NET");            //设置统计数据
$graph = new Graph(600, 300);                                   //设置画布大小
$graph->SetScale('textlin');                                    //设置坐标刻度类型
$graph->SetShadow();                                            //设置画布阴影
$graph->img->SetMargin(40, 30, 20, 40);                         //设置统计图边距
$barplot = new BarPlot($data);                                  //实例化BarPlat对象
$barplot->SetFillColor('blue');                                 //设置柱形图前景色
$barplot->value->Show();
$graph->Add($barplot);
$graph->title->Set(iconv("utf-8","gb2312","吉林省明日科技有限公司6月份编程词典销售量分析"));
                                                                    //统计图标题
$graph->xaxis->title->Set(iconv("utf-8","gb2312","部门"));          //设置X轴名称
$graph->xaxis->SetTickLabels($datas);
$graph->yaxis->title->Set(iconv("utf-8","gb2312",'总数量(本)'));    //设置Y轴名称
$graph->title->SetFont(FF_SIMSUN, FS_BOLD);                         //设置标题字体
$graph->xaxis->title->SetFont(FF_SIMSUN,FS_BOLD);                   //设置X轴字体
$graph->yaxis->title->SetFont(FF_SIMSUN,FS_BOLD);                   //设置Y轴字体
$graph->Stroke();                                                   //输出图像
```

运行效果如图 11.14 所示。

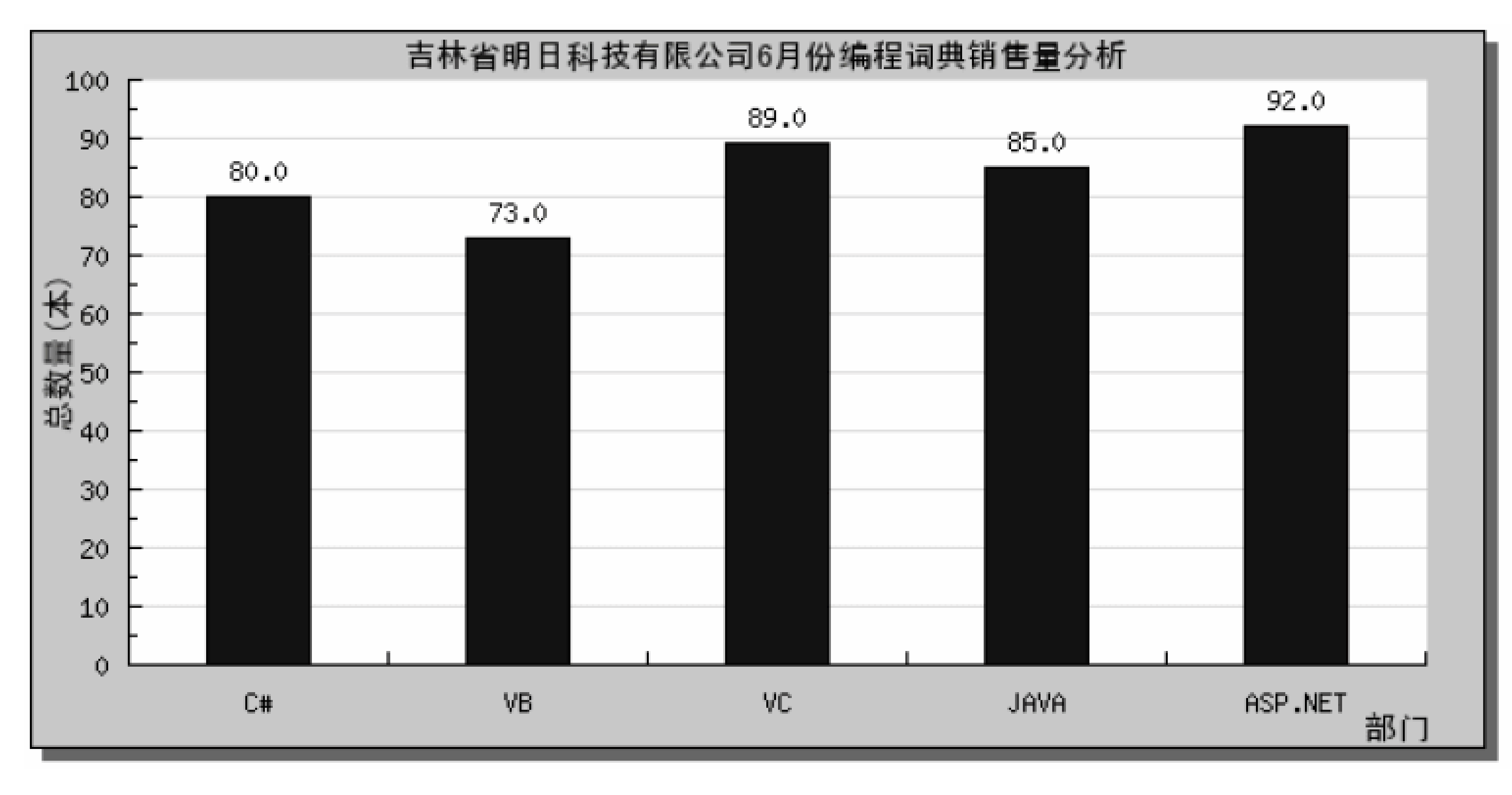


图 11.14　柱形图分析产品月销售量

11.4.4　折线图分析网站一天内的访问走势

例 11.7　通过 Jpgraph 类库创建折线图，对网站一天内的访问走势进行分析。其中，应用 SetFillColor 方法为图像填充颜色；通过 SetColor 方法定义数据、文字、坐标轴的颜色。其具体



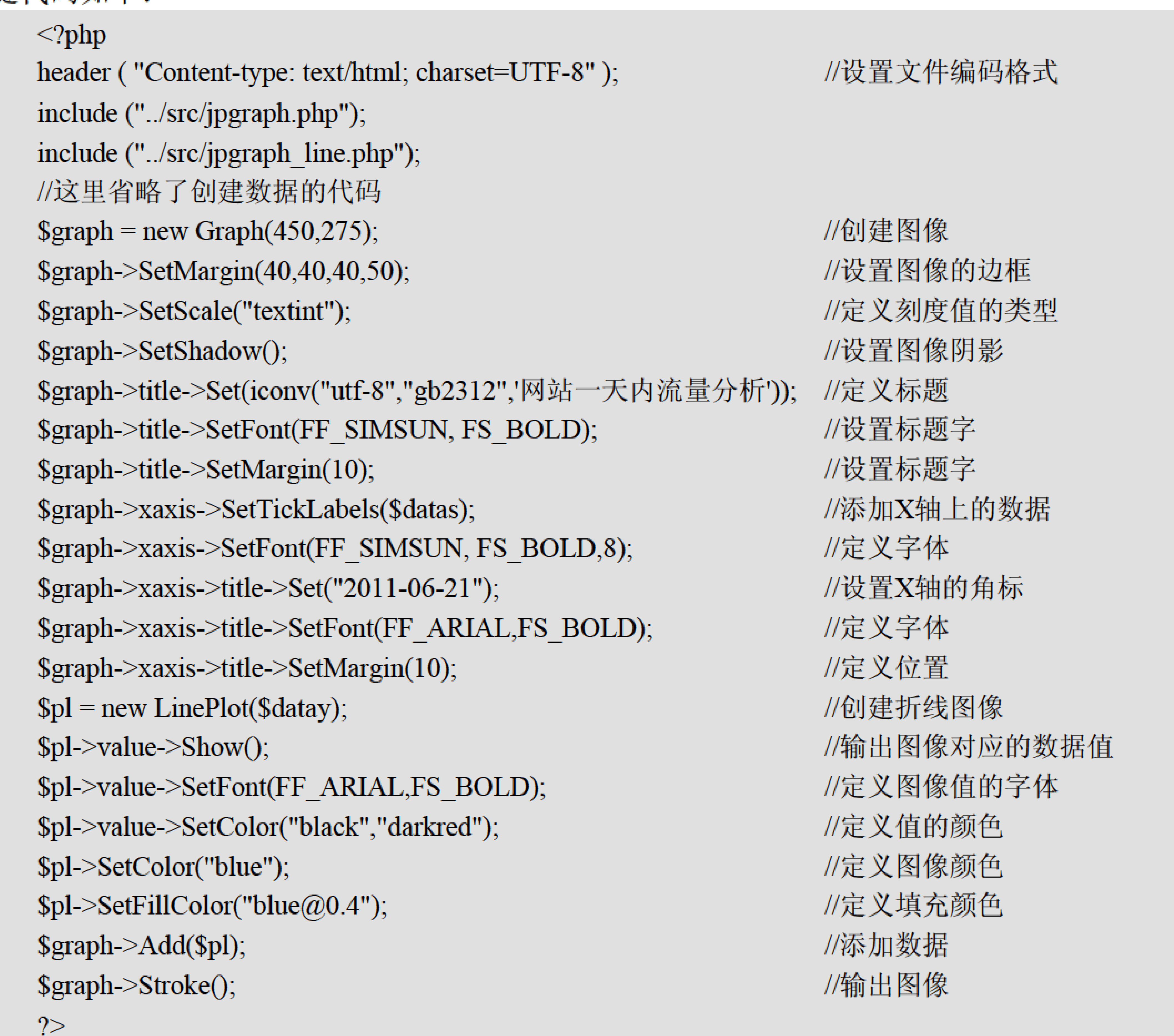




步骤如下。**（实例位置：配套资源\mr\11\example\11.7）**

（1）创建 index.php 文件，设置网页的编码格式，并通过 include()语句导入所需的存储在章文件夹 06 下的 Jpgraph 文件。注意，这里创建的是折线图，所以导入的文件也发生了变化。

（2）应用 Jpgraph 类库中的方法创建一个折线图，对网站一天中的访问量走势进行分析。其关键代码如下：



```
<?php
header ( "Content-type: text/html; charset=UTF-8" );                //设置文件编码格式
include ("../src/jpgraph.php");
include ("../src/jpgraph_line.php");
//这里省略了创建数据的代码
$graph = new Graph(450,275);                                        //创建图像
$graph->SetMargin(40,40,40,50);                                     //设置图像的边框
$graph->SetScale("textint");                                        //定义刻度值的类型
$graph->SetShadow();                                                //设置图像阴影
$graph->title->Set(iconv("utf-8","gb2312",'网站一天内流量分析'));   //定义标题
$graph->title->SetFont(FF_SIMSUN, FS_BOLD);                         //设置标题字
$graph->title->SetMargin(10);                                       //设置标题字
$graph->xaxis->SetTickLabels($datas);                               //添加X轴上的数据
$graph->xaxis->SetFont(FF_SIMSUN, FS_BOLD,8);                       //定义字体
$graph->xaxis->title->Set("2011-06-21");                            //设置X轴的角标
$graph->xaxis->title->SetFont(FF_ARIAL,FS_BOLD);                    //定义字体
$graph->xaxis->title->SetMargin(10);                                //定义位置
$pl = new LinePlot($datay);                                         //创建折线图像
$pl->value->Show();                                                 //输出图像对应的数据值
$pl->value->SetFont(FF_ARIAL,FS_BOLD);                              //定义图像值的字体
$pl->value->SetColor("black","darkred");                            //定义值的颜色
$pl->SetColor("blue");                                              //定义图像颜色
$pl->SetFillColor("blue@0.4");                                      //定义填充颜色
$graph->Add($pl);                                                   //添加数据
$graph->Stroke();                                                   //输出图像
?>
```

运行结果如图 11.15 所示。

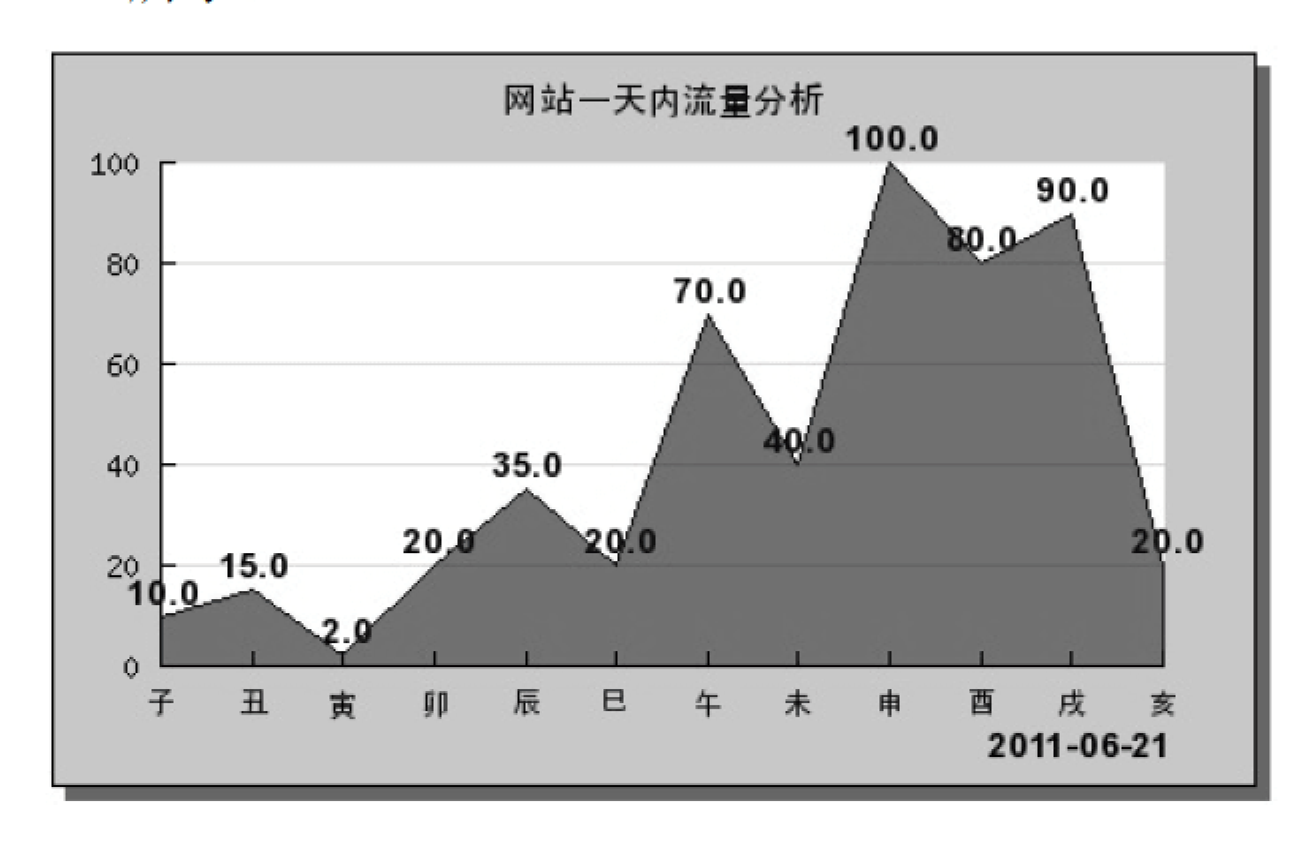


图 11.15　折线图分析网站一天内的访问走势





指点迷津：

在例 11.7 中以一天的 12 个时辰为单位，分析在这十二个时辰中网站访问量的走势。使用 Jpgraph 类库创建折线统计图，除了需要在程序中包含 jpgraph.php 文件外，还需要包含 jpgraph_line.php 文件，从而启用 Jpgraph 类库的折线创建功能。其中使用的 Jpgraph 技术如下：

（1）使用 LinePlot 对象绘制曲线。

通过 Jpgraph 类库中的 LinePlot 类创建曲线，该类的语法如下：

```
$linePlot = new LinePlot($data)        //创建折线图
```

参数$data：数值型数组，指定统计数据。

（2）使用 SetFont()方法统计图标题、坐标轴等文字样式。

制作统计图时，需要对图像的标题、坐标轴内文字进行样式设置，在 Jpgraph 类库中可以使用 SetFont()实现。该方法的语法如下：

```
SetFont($family, [$style,] [$size])
```

参数$family：指定文字的字体；参数$style：指定文字的样式；参数$size：指定文字的大小，默认为 10。

（3）使用 SetMargin()方法设置图像、标题、坐标轴上文字与边框的距离。其语法如下：

```
SetMargin($left,$right,$top,$bottom)
```

参数指定其与左右、上下边框的距离。

或者：

```
SetMargin($data)
```

参数$data 同样指定与边框的距离。

脚下留神：

创建不同的图像导入的文件是有所区别的。如果创建的是柱形图，那么导入的是：

```
include ("../src/jpgraph.php");
include ("../src/jpgraph_bar.php");
include ("../src/jpgraph_flags.php");
```

如果创建折线图，那么导入的是：

```
include ("../src/jpgraph.php");
include ("../src/jpgraph_line.php");
```

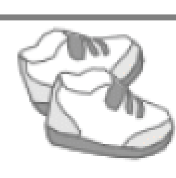
这点必须注意的是，如果没有导入正确的文件，那么就无法完成图像的创建操作。

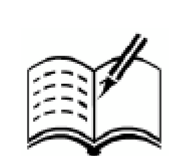
11.4.5　3D 饼形图展示各部门不同月份的业绩

例 11.8　Jpgraph 类库的制作统计图功能极其强大，使用该功能不仅可以绘制平面图形，而且可以绘制具有 3D 效果的图形。直接使用 GD2 函数库可以绘制出各种图形，当然也包括 3D 饼形图，但使用 GD2 函数绘制 3D 图形需要花费大量的时间，而且相对复杂，而采用 Jpgraph 类库绘制 3D 饼图却十分方便、快捷，具体实现过程如下。（**实例位置：配套资源\mr\11\example\11.8**）

（1）在程序中导入 Jpgraph 类库及饼图绘制功能，代码如下：

```
include ("../src/jpgraph.php");            //导入Jpgraph类库
include ("../src/jpgraph_pie.php");        //导入Jpgraph类库的饼图功能
include ("../src/jpgraph_pie3d.php");      //导入Jpgraph类库的3D饼形图功能
```





（2）创建数值型数组作为统计数据，代码如下。

```
$data = array(
    array(80,18,15,17),
    array(35,28,6,34),
    array(10,28,10,5),
    array(22,22,10,17));                          //设置统计数据
```

（3）创建统计图对象，并对统计图的标题内容、字体进行设置，代码如下。

```
$graph->title->Set(iconv('utf-8','gbk',"部门业绩比较"));      //设置标题
$graph->title->SetFont(FF_SIMSUN,FS_BOLD,20);              //设置字体
$graph->title->SetColor('white');                          //设置颜色
```

（4）创建 3D 饼图形对象并输入统计图，代码如下。

```
$titles = array('C#','.net','JAVA','PHP');
$n = count($piepos)/2;
$graph = new PieGraph(550,400,'auto');
// Specify margins since we put the image in the plot area
$graph->SetMargin(1,1,40,1);
$graph->SetMarginColor('navy');
$graph->SetShadow(false);
// Setup background
$graph->SetBackgroundImage('worldmap1.jpg',BGIMG_FILLPLOT);
```

运行结果如图 11.16 所示。

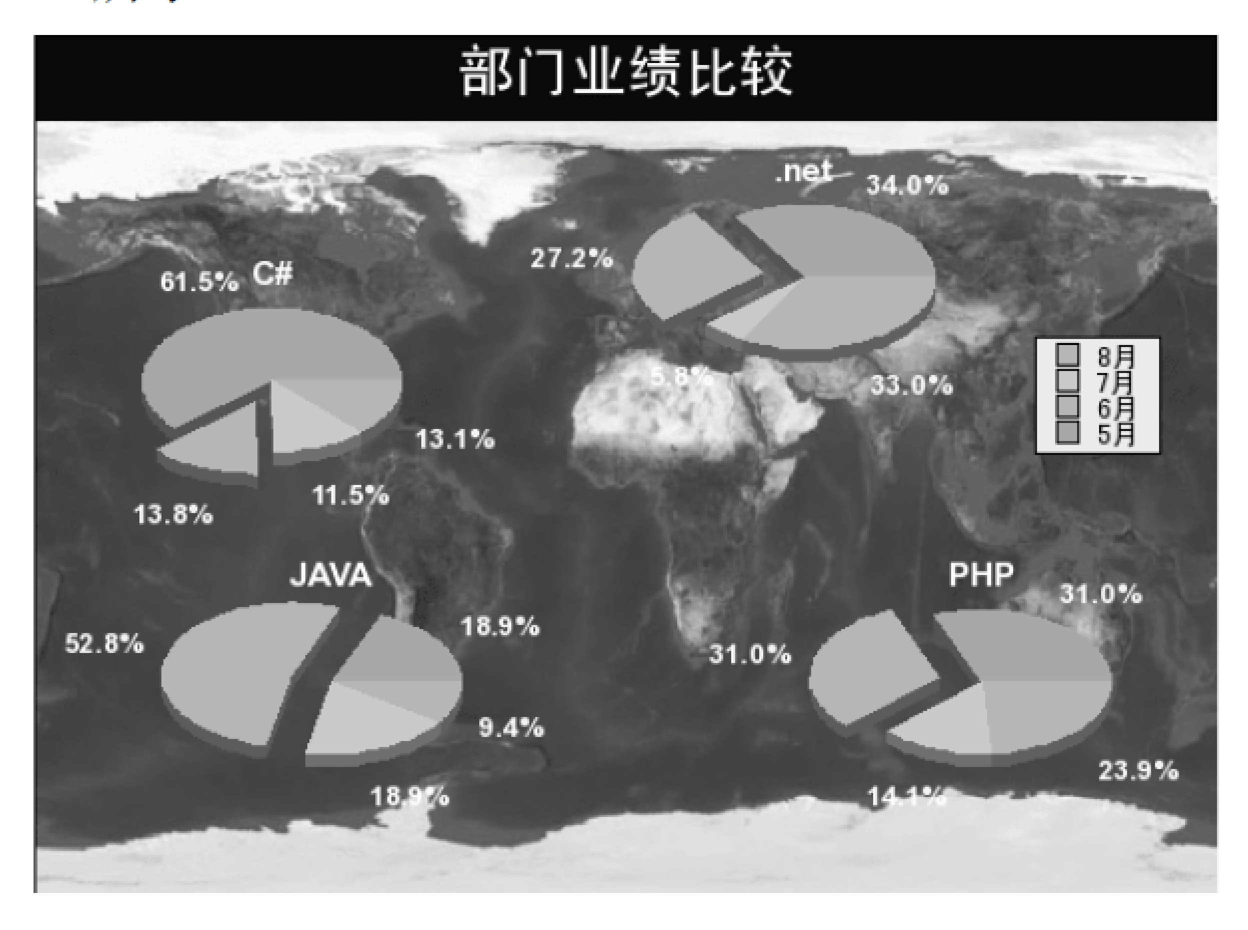


图 11.16　部门业绩比较

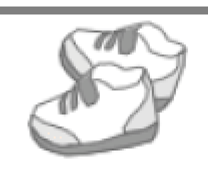
上机演练

上机演练 5　折线图分析 2010 年销售额

运用 Jpgraph 类库生成折线图，对公司 2010 年的销售额进行分析，其运行结果如图 11.17 所示。

上机演练 6　柱形图分析编程词典销售比例

运用 Jpgraph 类库生成柱形图，对编程词典的销售比例进行分析，其运行结果如图 11.18 所示。



上机演练 7　饼形图分析 2010 年图书销量

运用 Jpgraph 类库生成饼形图，对公司 2010 年图书销量进行分析，其运行结果如图 11.19 所示。

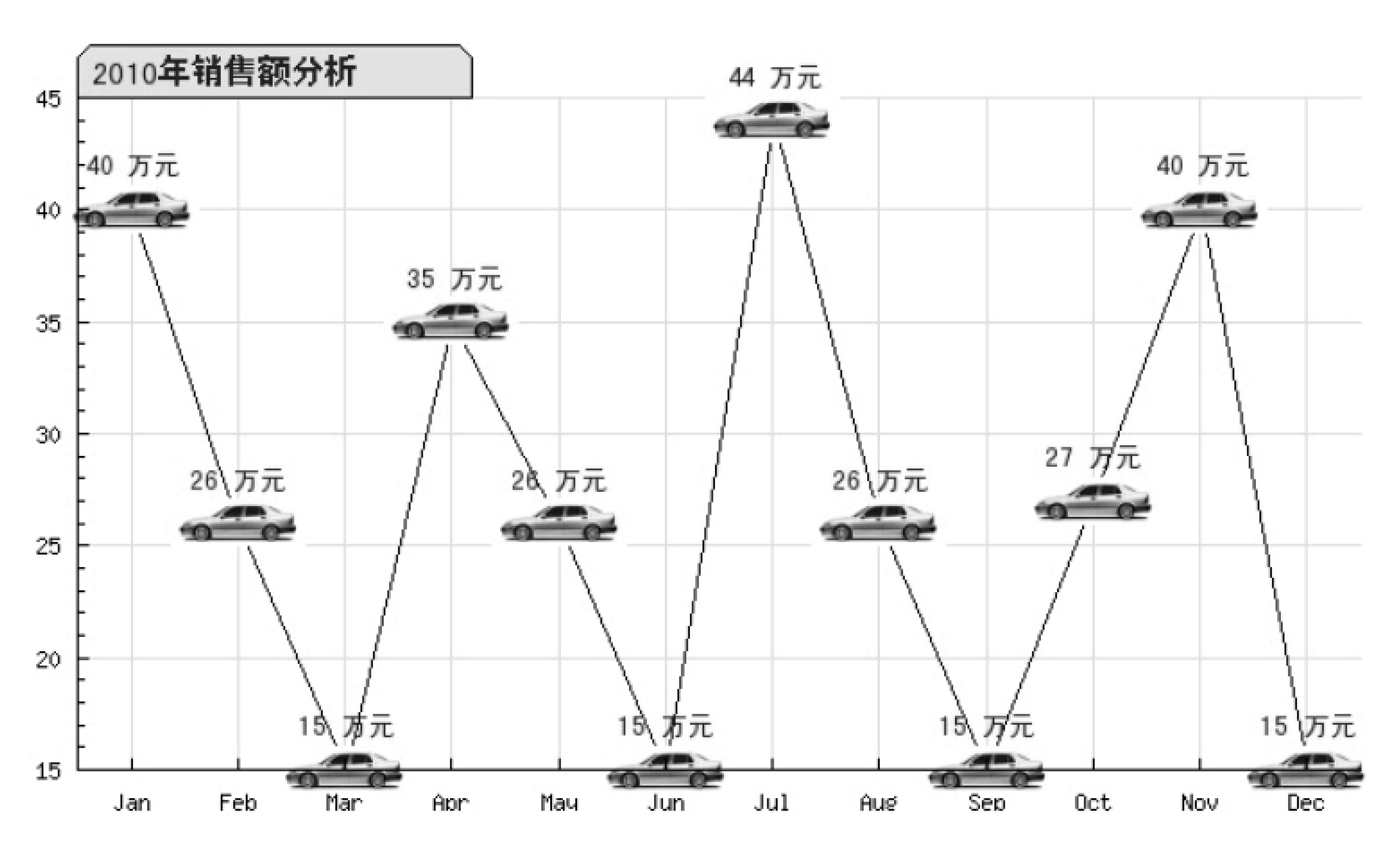


图 11.17　公司 2010 年销售额分析

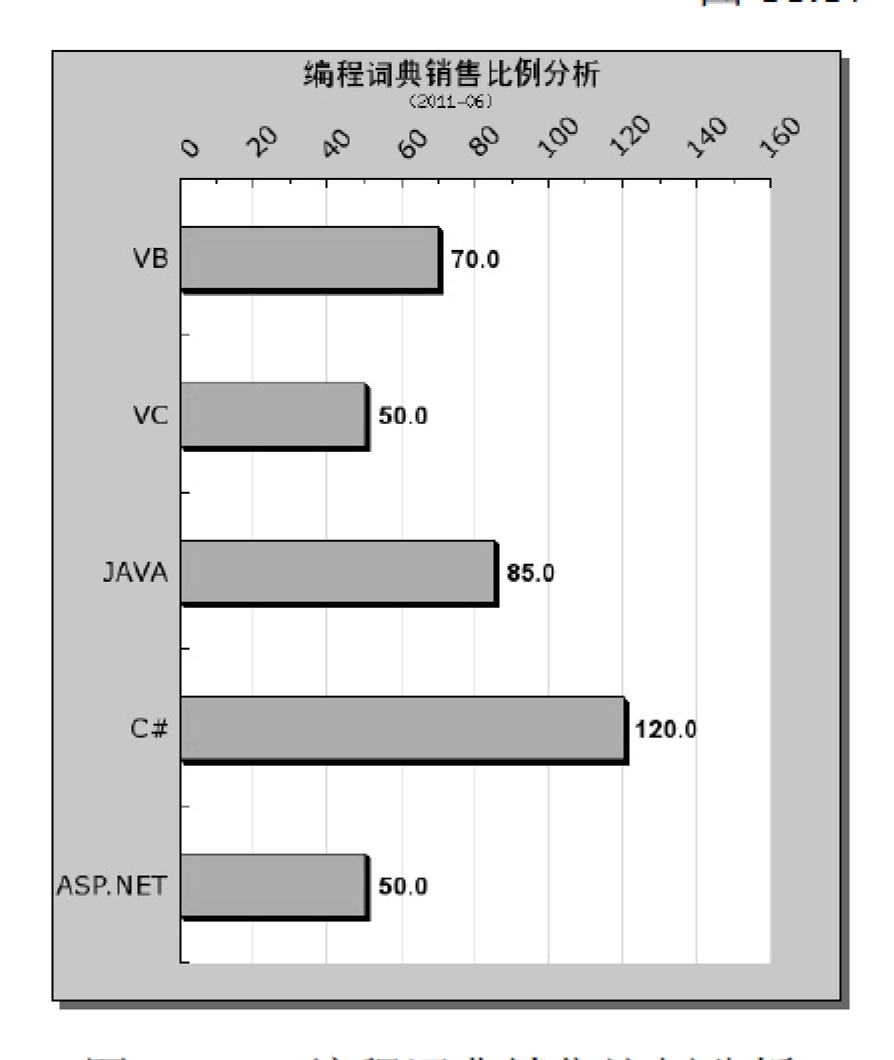


图 11.18　编程词典销售比例分析

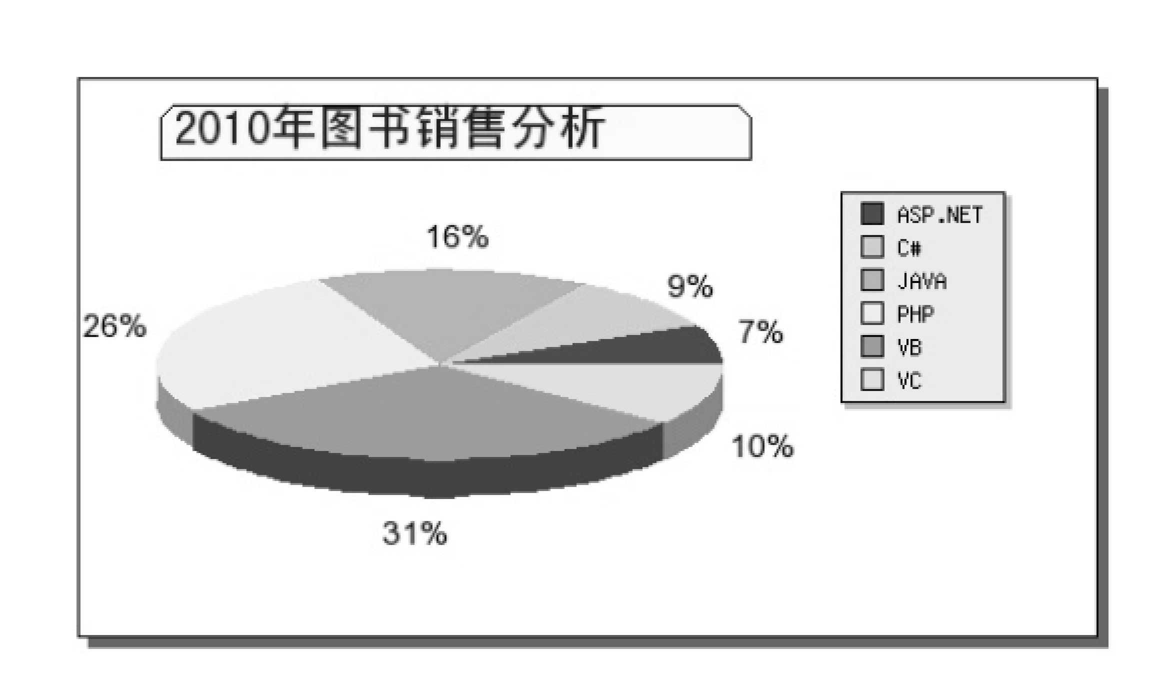


图 11.19　公司 2010 年图书销量分析

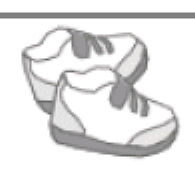
本 章 摘 要

1．通过 GD2 函数库创建各种图形图像，包括输出中、英文字符串，向图片中添加水印，生成图像验证码以及创建各种统计图形。

2．Jpgraph 类库的应用，包括绘制柱形图、折线图以及 3D 饼形图等。

习　　题

1．Jpgraph 是使用（　　）编程语言编写的。



Note

A．c　　　B．java　　　C．php

2．如何判断 PHP 是否支持 GD2 函数库（　　）。

A．连接数据库　　　B．浏览 phpinfo.php 页面　　　C．浏览 php.ini 文件

3．Rand 函数的主要用途是（　　）。

A．获取随机数　　　B．转换字符串　　　C．加密操作

4．告知浏览器页面输出的是一个图像需要使用（　　）函数。

A．rand　　　B．microtime　　　C．header

5．看下面一行代码：

```
Header("Content-Type:image/jpeg");
```

此代码告知浏览器输出图像的格式为（　　）。

A．GIF　　　B．PNG　　　C．JPEG

6．完成代码，使其输出 800×800 的图像。

```
header("Content-Type:image/jpeg");
 (                                        )
Imagejpeg($image);
```

7．销毁图像需要使用（　　　　）函数。

8．创建画布需要使用（　　　　）函数。

9．GD2 函数中绘制中文文字需要使用（　　　　　　）函数。

10．程序运行后，显示的图片长宽分别为（　　）、（　　）。

```
<?php
     header("Content-Type:image/jpeg");
     $image=imagecreate(130,30);
     $bgcolor=imagecolorallocate($image,230,230,230);
     $fontcolor=imagecolorallocate($image,70,70,70);
     imagestring($image,5,16,6,"A B C D E F",$fontcolor);
     imagejpeg($image);
     imagedestroy($image);
?>
```

实战模拟

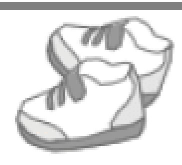
学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会。（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 10 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　应用 GD2 函数绘制折线图分析网站月访问量走势

应用 GD2 函数自行编写一个绘制折线图的方法，对 2010 年某公司网站的月访问量进行分析。该方法完全由笔者自行编写，不借助于任何图像的操作类库。其运行结果如图 11.20 所示。

实战模拟 2　应用 GD2 函数绘制柱形图分析编程词典满意度调查

在网站中，经常需要对相关的信息进行调查统计，然后根据访问者的投票结果制定相关计划，为了能够更直观地查看访问者的投票结果，采用柱形图显示编程满意度的投票结果是一个不错的方法。其运行结果如图 11.21 所示。



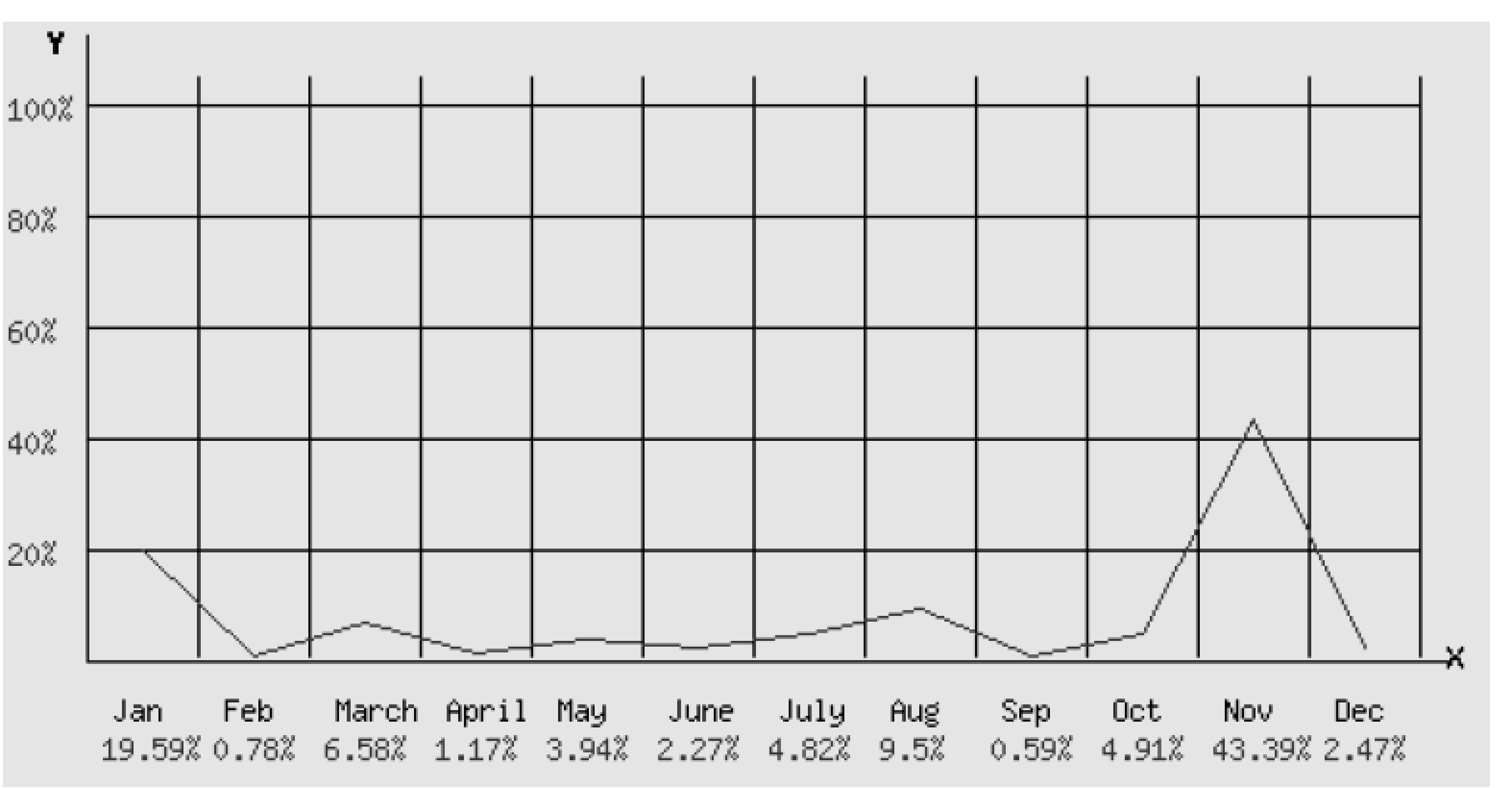


图 11.20　应用 GD2 函数绘制折线图分析网站月访问量走势

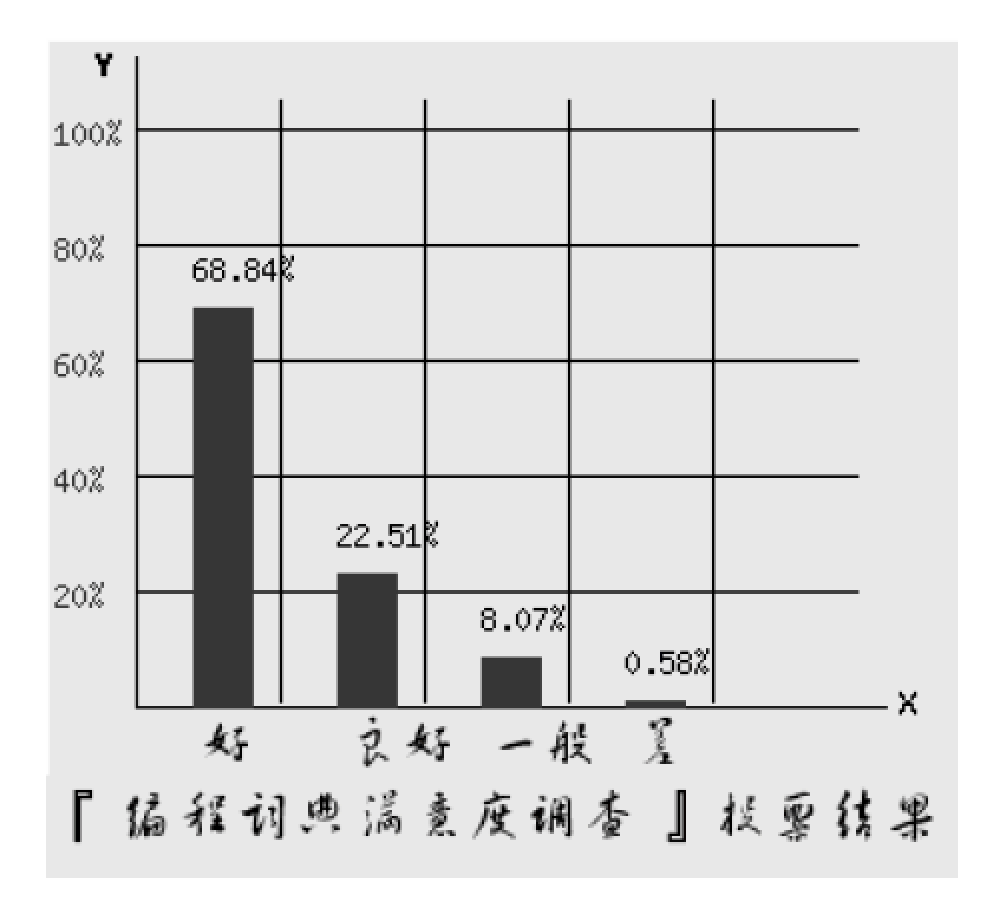


图 11.21　应用 GD2 函数绘制柱形图分析编程词典满意度调查

实战模拟 3　应用 GD2 函数绘制饼形图分析图书市场的份额

在调查某类商品的市场占有率时，最好的显示方式就是使用饼形图。通过饼形图可以直观地看到某类产品的不同品牌在市场中的占有比例。本例通过饼形图，将不同语言的软件图书在市场中的占有率显示出来，运行结果如图 11.22 所示。

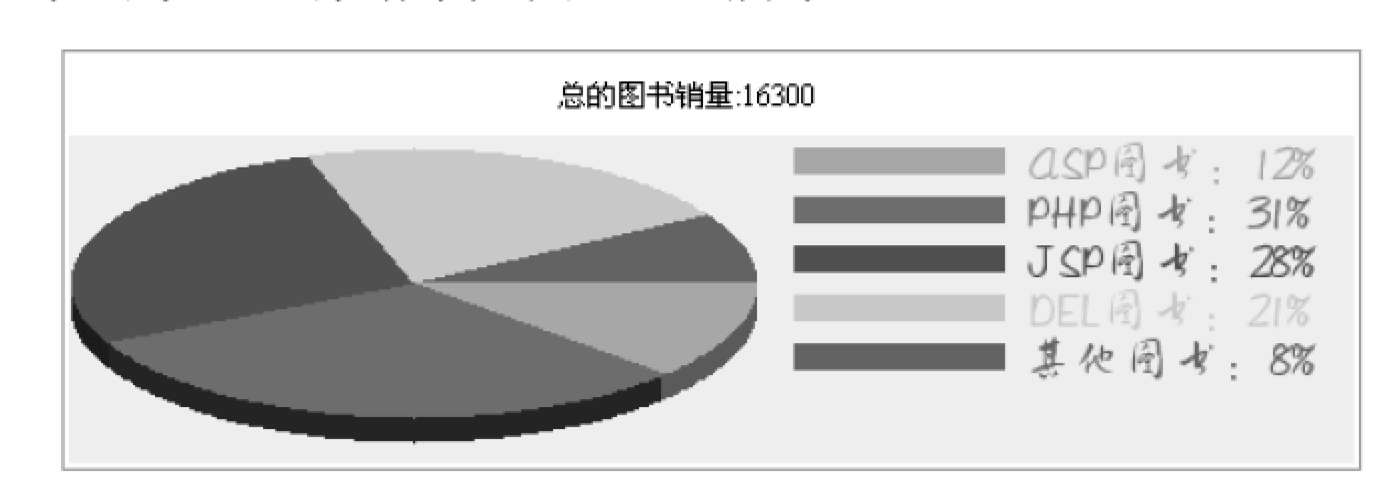


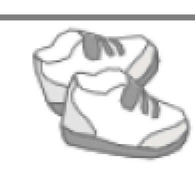
图 11.22　应用 GD2 函数绘制饼形图分析软件图书市场占有率

实战模拟 4　柱形图展示编程词典 6、7 月份销售量

运用 Jpgraph 类库生成对比柱形图，对编程词典 6、7 月份的销售量进行比较，其运行结果如图 11.23 所示。

实战模拟 5　柱形图展示编程词典上半年销量

运用 Jpgraph 类库生成柱形图，对公司编程词典上半年的销量进行统计，其运行结果如



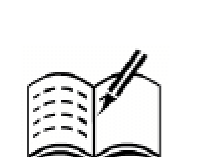




图 11.24 所示。

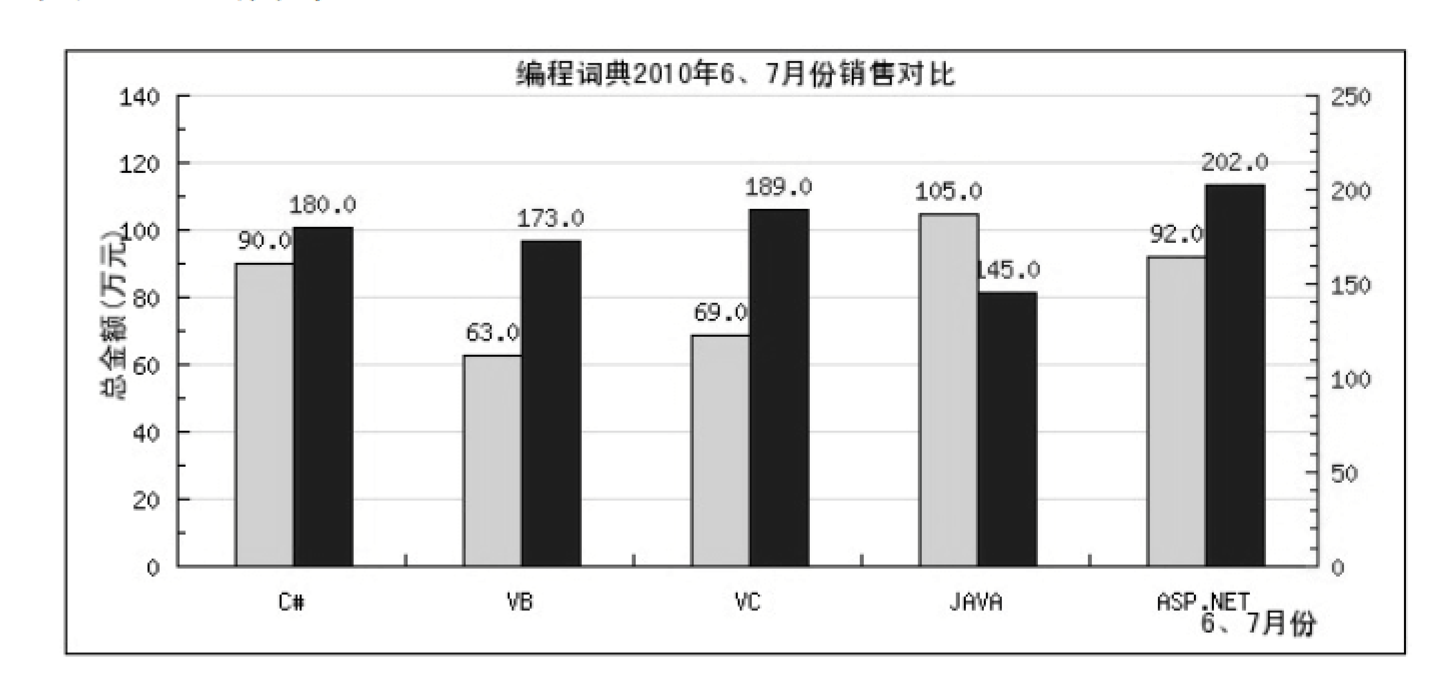


图 11.23　柱形图展示编程词典 6、7 月份销量

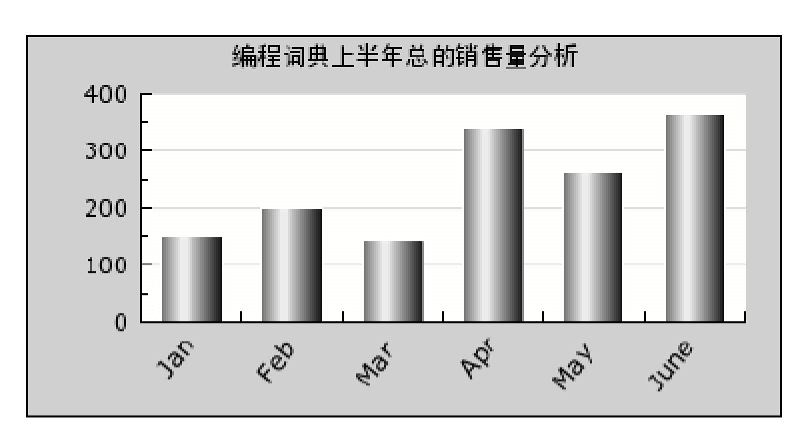


图 11.24　柱形图展示编程词典上半年销量

实战模拟 6　折线图分析 2010 年牛肉市场价格走势

运用 Jpgraph 类库生成折线图，对 2010 年牛肉市场的价格走势进行分析，其运行结果如图 11.25 所示。

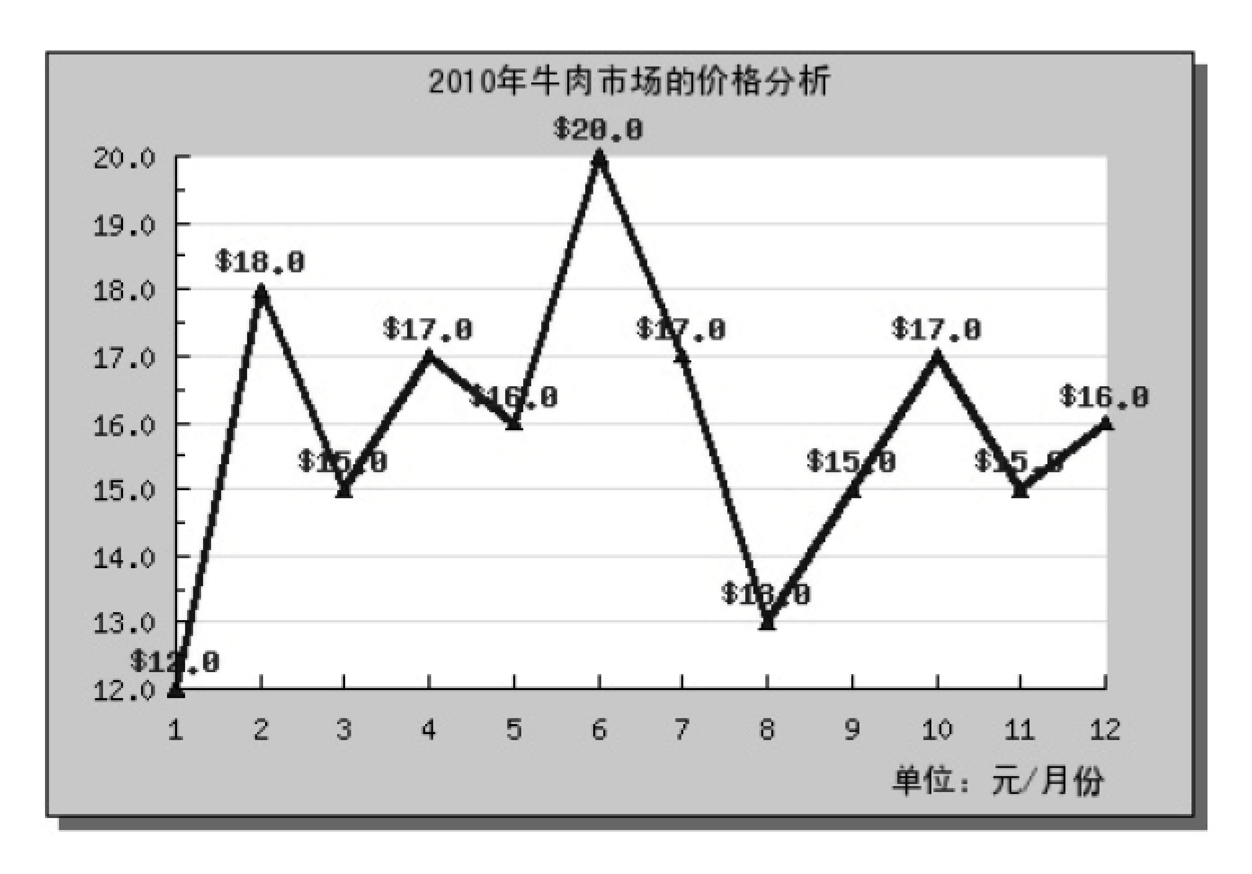


图 11.25　折线图分析 2010 年牛肉市场价格走势

实战模拟 7　多饼形图区块分析 2010 年图书销量

运用 Jpgraph 类库生成饼形图，对公司 2010 年图书销量进行分析，其运行结果如图 11.26 所示。

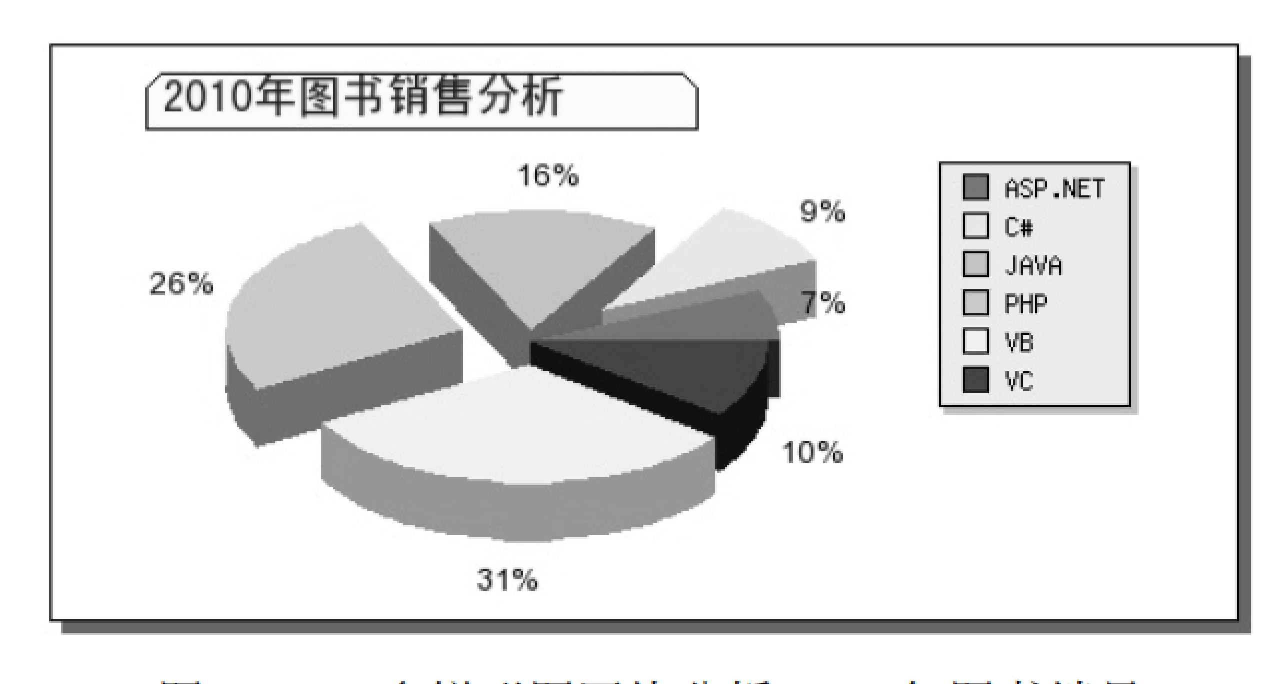


图 11.26　多饼形图区块分析 2010 年图书销量

实战模拟 8　缩略图艺术库

上传图片是一个比较常用的功能，但是在输出上传图片时可能会遇到一些问题，如由于上



传图片的大小不固定，而导致图片在输出时发生变形，这是一个很伤脑筋的问题。如果在上传图片时直接将其生成一个固定大小的缩略图，并同时保存上传的原始图片，那么在输出时就不会有任何问题。

这里就实现这样一个功能，在将图片上传到服务器的同时生成图片的缩略图，直接浏览图片的缩略图即可，而如果要进行下载，则下载的仍是原始图片。运行效果如图 11.27 所示。

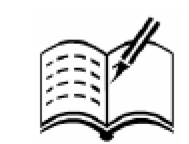




图 11.27　缩略图艺术库

第12章

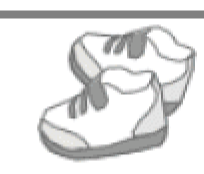
文件、目录处理

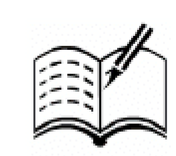
（自学视频、源程序：配套资源\mr\12\）

掌握文件处理技术对于 Web 开发者来说是十分必要的。虽然在处理信息方面，使用数据库是多数情况下的选择，但对于少量的数据，利用文件来存取是非常方便快捷的，更关键的是 PHP 中提供了非常简单方便的文件、目录处理方法。在本章中将对 PHP 操作文件和目录的技术进行系统讲解。

学习摘要：

- 打开一个文本文件
- 读取文件内容
- 向文件中写入数据
- 文件的创建、复制、移动和删除等技术
- 关闭文件指针
- 打开指定目录
- 读取目录结构
- 关闭目录指针





12.1　基本的文件处理

视频讲解：配套资源\mr\12\video\基本的文件处理.exe

文件操作是通过 PHP 内置的文件系统函数完成的。文件和目录操作可以归纳为如下 3 个步骤：

（1）打开文件的方法。

（2）读取、写入和操作文件的方法。

（3）关闭文件的方法。

用户若把握住上述 3 个步骤，就可以区分出函数使用的先后顺序和功能，做到运用自如。

12.1.1　打开一个文件

要对文件进行操作，首先必须要打开这个文件。

在 PHP 中使用 fopen()函数打开一个文件，语法如下：

```
resource fopen (string filename, string mode [, int use_include_path [, resource zcontext]])
```

参数 filename 指定打开的文件名。

指点迷津：

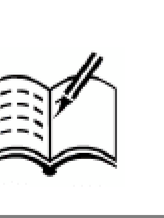
参数 filename 可以是包含文件路径的文件名（例如 C:/windows/php.ini 或./php.ini）。为了避免不同系统之间切换可能带来的麻烦，采用“/”作为路径分隔符。另外，参数 filename 也可以是由某种协议给出的 URL（例如，http://mrbccd.cn 或 ftp://www.mrbccd.cn//）。如果指定 URL 地址，则可以打开远程文件。

参数 mode 设置打开文件的方式，参数值如表 12.1 所示。

表 12.1　fopen()函数中参数 mode 的参数值

mode	模 式 名 称	说　明
r	只读	读模式——读文件，文件指针位于文件头部
r+	只读	读写模式——读、写文件，文件指针位于文件头部。注意：如果在现有文件内容的末尾之前进行写入就会覆盖原有内容
w	只写	写模式——写入文件，文件指针指向文件头部。注意：如果文件存在，则文件内容被删除，重新写入；如果文件不存在，则函数自行创建文件
w+	只写	写模式——读、写文件，文件指针指向文件头部。注意：如果文件存在，则文件内容被删除，重新写入；如果文件不存在，则函数自行创建文件
x	谨慎写	写模式打开文件，从文件头部开始写入。注意：如果文件已经存在，则函数返回 False，并产生一个 E_WARNING 级别的错误信息
x+	谨慎写	读/写模式打开文件，从文件头部开始写入。注意：如果文件已经存在，则函数返回 False，并产生一个 E_WARNING 级别的错误信息
a	追加	追加模式打开文件，文件指针指向文件尾部。注意：如果文件已有内容，则将从文件末尾开始追加；如果文件不存在，则函数自行创建文件
a+	追加	追加模式打开文件，文件指针指向文件尾部。注意：如果文件已有内容，则从文件末尾开始追加或读取；如果文件不存在，则函数自行创建文件





续表

mode	模 式 名 称	说　　明
b	二进制	二进制模式——用于与其他模式进行连接。注意：如果文件系统能够区分二进制文件和文本文件，可能会使用它。Windows 可以区分，UNIX 则不区分。推荐使用这个选项，便于获得最大程度的可移植性。它是默认模式
t	文本	用于与其他模式的结合。该模式只是 Windows 下的一个选项

参数 use_include_path 为可选参数，决定是否在 include_path（php.ini 中的 include_path 选项）定义的目录中搜索 filename 文件。例如，在 php.ini 文件中设置 include_path 选项的值为“E:\AppServ\www\MR\Instance\10\”，如果希望服务器在该路径下打开所指定的文件，则设置参数 use_include_path 的值为 1 或 True。

参数 context 称为上下文，同样为可选参数，是设置流操作的特定选项，用于控制流的操作特性。一般情况下只需使用默认的流操作设置，而无须使用此参数。

多学两招：

在使用 fopen()函数时应十分谨慎，因为从该函数的参数中可以看到，使用该函数不像平时使用用 Note、Word 那么简单，一不小心就有可能将文件内容全部删掉。

例 12.1 通过 fopen()函数打开指定的文件，代码如下：（**实例位置：配套资源\mr\12\example\12.1**）

```
<?php
//以只读方式打开当前执行脚本所在目录下的count.txt文件
$file1=fopen("./count.txt","r");
//以读写方式打开指定文件夹下的文件，如果文件不存在，则创建
$file2=fopen("C:/count.txt","w+");
//以二进制只写方式打开指定文本，并清空文件
$file3=fopen("./images/bg_01.jpg","wb");
//以只读方式打开远程文件
$file4=fopen("http://192.168.1.249/in/index.php","r");
?>
```

运行本实例，在页面中看不到任何效果。但此时可以查看一下 C 盘的根目录，会发现有一个 count.txt 文件。在本实例根目录下的 images 文件夹中存储了两个图片：一个是 bg_01.jpg，另一个是 bg_01.jpg 的备份，读者可以比较一下它们的不同。

另外，如果用户在本实例的根目录下未创建 count.txt 文件，那么运行本程序后将看到如图 12.1 所示的错误信息。

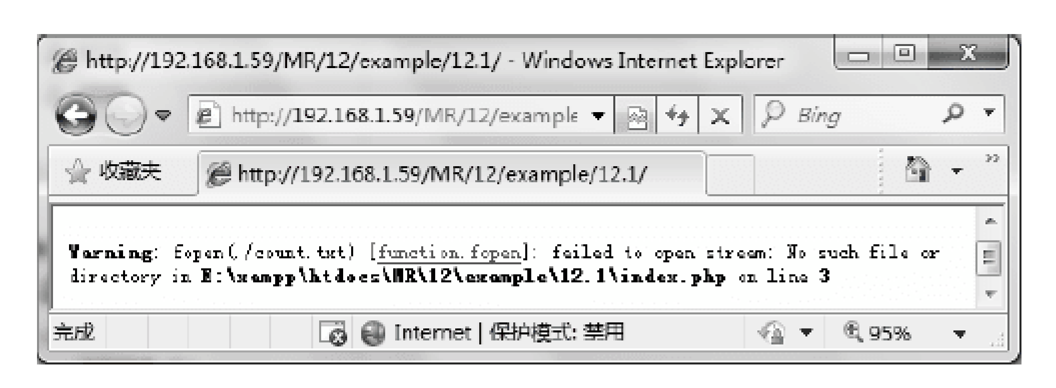
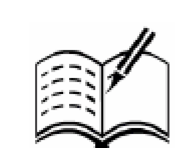


图 12.1　没有找到指定文件，返回错误信息

12.1.2　读取文件内容

文件打开之后，就可以进行读取和写入操作了，这里先讲解文件的读取。可以将 PHP 提





供的文件读取函数分为 4 类：读取一个字符、读取一行字串、读取任意长度的字串和读取整个文件。

1. fgetc()函数（读取一个字符）

fgetc()函数从文件指针指定的位置读取一个字符。函数语法如下：

```
string fgetc ( resource handle )
```

该函数返回一个字符，该字符从 handle 指向的文件中得到。若遇到 EOF 则返回 False。

例 12.2 应用 fgetc()函数循环读取 in.txt 文件中的字符。**（实例位置：配套资源\mr\12\example\12.2）**

（1）在当前目录下创建 txt 文件夹，新建一个 in.txt 文本文件。向文件中写入如下内容，然后保存并关闭文件。

```
<font size="13" color="red">hello world</font>
```

（2）创建 index.php 文件。首先定义文件路径，然后用只读方式打开文件，由于 fgetc()函数只能读取单个字符，所以为了拼接 for 循环的循环条件，这里使用 filesize()函数判断文本文件的数据长度。最后利用 fgets()函数输出文本数据。核心代码如下：

```
<?php
header("Content-Type:text/html; charset=gb2312");          //设置页面编码格式
$path="txt/in.txt";                                         //定义文件路径
$fopen=fopen($path,"r");                                    //打开指定文件（只读方式）
$size=filesize($path);                                      //获取文件的数据长度
for($a=0;$a<$size;$a++){                                    //for循环语句
    echo fgetc($fopen);                                     //输出数据
}
fclose($fopen);                                             //关闭文件
?>
```

运行结果如图 12.2 所示。

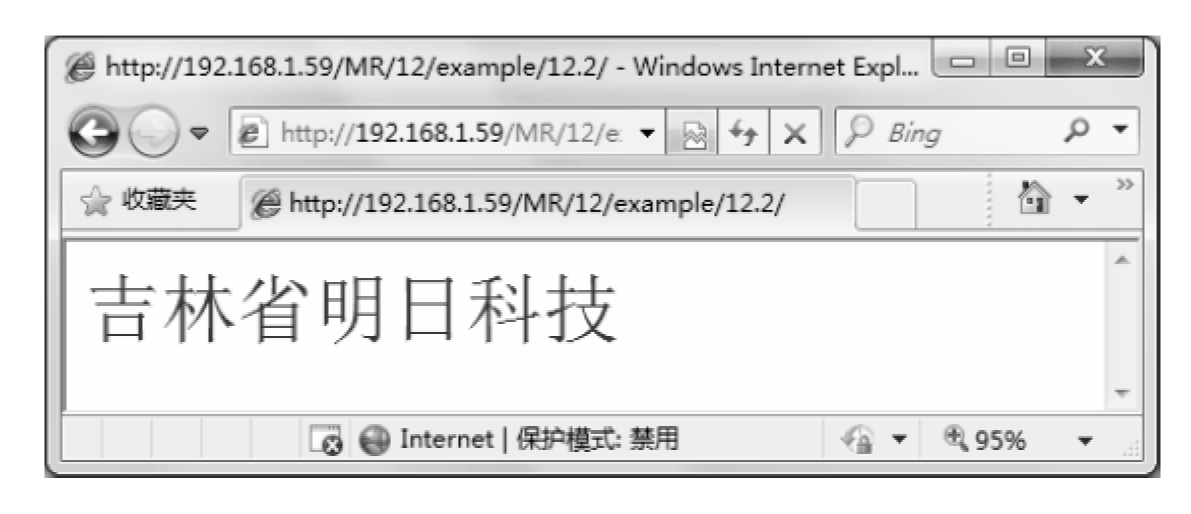


图 12.2 循环读取文件数据

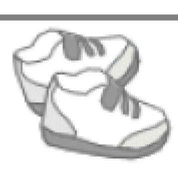
2. fgets()函数（读取一行字符）

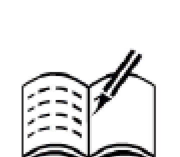
fgets()函数从文件指针中读取一行数据。需要注意的是，此时的文件指针必须是有效的，且必须指向一个由 fopen()或 fsockopen()函数成功打开的文件。fgets()函数语法如下：

```
string fgets( int handle [, int length] )
```

参数 handle 是被打开的文件；参数 length 是要读取的数据长度。

fgets()函数能够从 handle 指定文件中读取一行并返回长度最大值为 length-1 个字节的字符串。在遇到换行符、EOF 或读取了 length-1 个字节后停止。如果忽略 length 参数，那么将读取到行结束。





多学两招：

fgetss()函数是 fgets()函数的变体，用于读取一行数据，同时 fgetss()函数会过滤掉被读取内容中的 html 和 php 标记。语法如下：

string fgetss (resource handle [, int length [, string allowable_tags]])

参数 handle 指定读取的文件；参数 length 指定读取字符串的长度；参数 allowable_tags 控制哪些标记不被去掉。

例 12.3 应用 fgets()和 fgetss()两个函数分别输出文本文件的内容，看两者之间有什么区别。**（实例位置：配套资源\mr\12\example\12.3）**

```
<?php
    $fopen = fopen('./files.php','rb');
    while(!feof($fopen)){                    //feof()函数测试指针是否到了文件结束的位置
        echo fgets($fopen);                  //输出当前行
    }
    fclose($fopen);
?>
<!--  fgetss()函数读取.php文件  -->
<?php
    $fopen = fopen('./files.php','rb');
    while(!feof($fopen)){                    //使用feof测试指针是否到了文件结束的位置
        echo fgetss($fopen);                 //输出当前行
    }
    fclose($fopen);
?>
```

运行结果如图 12.3 所示。

从图 12.3 中可以看出，应用 fgets()函数读取的数据原样输出，没有任何变化；而应用 fgetss()函数读取的数据，去除了文件中的 html 标记，输出的完全是普通字符串。

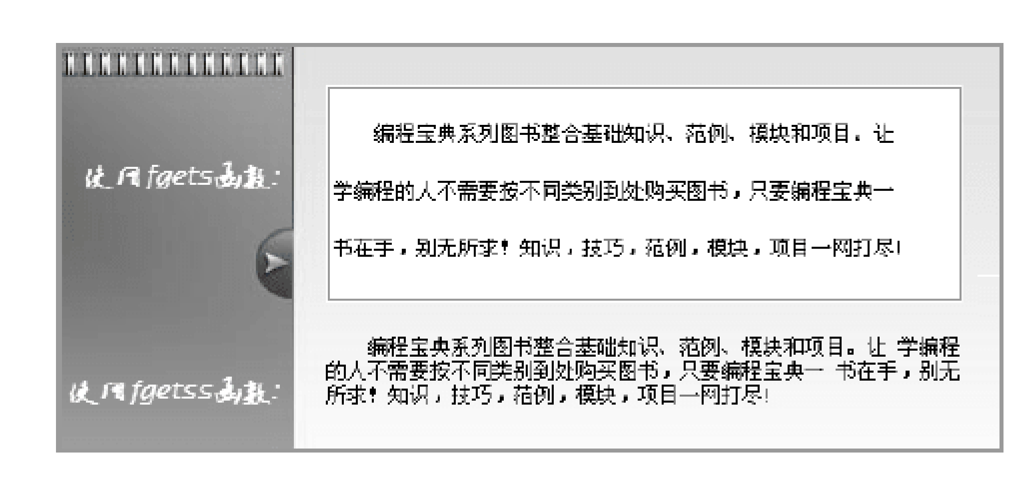


图 12.3 比较 fgets()和 fgetss()函数的输出结果

3. fread()函数（读取任意长度的字符串）

fread()函数从文件中读取任意长度的数据，还可以用于读取二进制文件。函数语法如下：

```
string fread ( int handle, int length )
```

参数 handle 为指向的文件资源；参数 length 指定要读取的字节数。此函数在读取到 length 个字节或到达 EOF 时停止执行。

例 12.4 应用 fread()函数读取 txt 文件夹下 in.txt 文件的内容，具体步骤如下。**（实例位置：配套资源\mr\12\example\12.4）**

首先，定义文本文件在实例根目录下的存储位置。

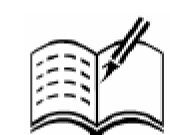
其次，利用 fopen()函数以只读方式打开文件并返回文件句柄。

然后，利用 filesize()函数获取文本文件数据的长度。

最后，利用 fread()含函数输出文本数据。代码如下：

```
<?php
    $path="txt/in.txt";                  //定义文本文件路径
```





```
        $open=fopen($path,"r");             //打开指定文件
        $size=filesize($path);              //获取文本数据长度
        echo fread($open,$size);            //输出所有数据
        fclose($open);                      //关闭文件
    ?>
```

运行结果如图 12.4 所示。

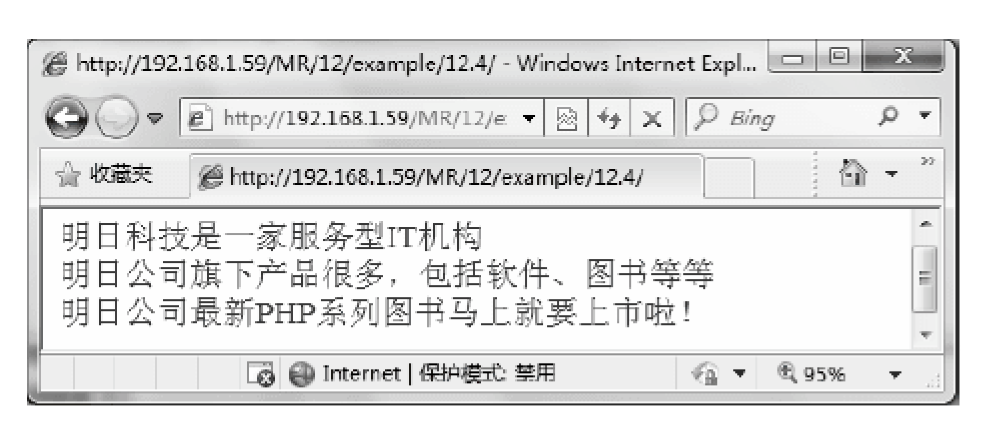


图 12.4　fread()函数的使用

4. readfile()、file()和 file_get_contents()函数（读取整个文件）

☑　readfile()函数

readfile()函数读取一个文件并写入到输出缓冲，成功则返回读取的字节数，失败则返回 False。语法如下：

```
int readfile ( string filename [, bool use_include_path [, resource context]] )
```

参数 filename 指定读取的文件名称；参数 use_include_path 控制是否支持在 include_path 中搜索文件，如果支持，则将该值设置为 True；参数 context 是 PHP5.0 的新增内容。

指点迷津：

应用 readfile()函数，不需要打开/关闭文件，也不需要输出语句，只需直接应用函数即可。

☑　file()函数

file()函数将整个文件的内容读入到一个数组中。成功返回数组，数组中的每个元素都是文件中对应的一行，包括换行符在内；失败返回 False。语法如下：

```
array file ( string filename [, int use_include_path [, resource context]] )
```

其参数与 readfile()函数相同，唯一的区别是该函数的返回值是数组。

☑　file_get_contents()函数

file_get_contents()函数将文件内容读入一个字符串。如果有 offset 和 maxlen 参数，则在参数 offset 所指定的位置开始读取长度为 maxlen 的内容。如果失败，则返回 False。语法如下：

```
string file_get_contents ( string filename [, bool use_include_path [, resource context [, int offset [, int maxlen]]]] )
```

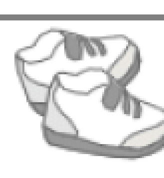
参数 filename 指定读取的文件名称；参数 use_include_path 控制是否支持在 include_path 中搜索文件，如果支持，则将该值设置为 True。

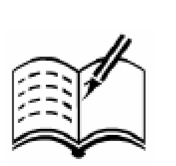
多学两招：

读取整个文件中的内容，推荐读者使用 file_get_contents()函数。

例 12.5　在了解了上述函数之后，下面编写一个实例，体会一下这些函数的功能。读取指定文件中的全部内容。（**实例位置：配套资源\mr\12\example\12.5**）

创建 index.php 文件，分别应用 readfile()、file()和 file_get_contents()函数读取指定文件中的





Note

内容。代码如下：

```
<?php
function type($number,$path="txt/in.txt"){              //自定义函数，将path参数定义为可选参数
    if($number=="1"){                                     //判断传递进来的参数是否等于1
        echo '<h2>file_get_contents()输出数据</h2>';      //输出数据
        echo file_get_contents($path);
    }else{
        if($number=="2"){                                 //判断传递进来的参数是否等于2
            echo '<h2>readfile()输出数据</h2>';           //输出数据
            readfile($path);
        }else{
            $array=file($path);                           //将数据保存在数组中
            echo '<h2>file()输出数据</h2>';               //循环输出数据
            for($a=0;$a<count($array);$a++){
                echo "#".$array[$a]."<br>";
            }
        }
    }
}
type("3");                                                //方法调用
type("2");
type("1");
?>
```

运行结果如图 12.5 所示。

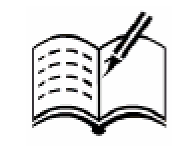


图 12.5　读取整个文件的输出结果

脚下留神：

不知道大家是否已经注意到，在通过 readfile()、file()和 file_get_contents()函数读取整个文件中的内容时，不需要通过 fopen()函数打开文件，也不需要使用 fclose()函数关闭文件。但是，在读取一个字符、读取一行字符和读取任意长度的字符串时必须应用 fopen()函数打开文件后才能进行读取，在读取完成后还要应用 fclose()函数关闭文件。





12.1.3 向文件中写入数据

前面介绍了文件的打开和读取操作，下面介绍文件的写入操作。PHP 中通过 fwrite()和 file_put_contents()函数执行文件的写入操作。

1. fwrite()函数（向文件中写入数据）

fwrite()函数执行文件的写入操作。另外，它还有一个别名为 fputs()。其语法如下：

```
int fwrite ( resource handle, string string [, int length] )
```

fwrite()函数把 string 的内容写入文件指针 handle 处。如果设置 length，那么当写入 length 个字节或完成 string 的写入后，操作就会停止。若 fwrite()函数操作成功，则返回写入的字符数，失败则返回 False。

多学两招：

在应用 fwrite()函数时，如果给出 length 参数，那么 magic_quotes_runtime（php.ini 文件中的选项）配置选项将被忽略，而 string 中的斜线将不会被抽去。如果在区分二进制文件和文本文件的系统上（例如 Windows）应用这个函数，当打开文件时，fopen()函数的 mode 参数要加上'b'。

2. file_put_contents()函数（向文件中写入数据）

file_put_contents()函数将一个字符串写入文件中。若操作成功，则返回写入的字节数，失败则返回 False。其语法如下：

```
int file_put_contents ( string filename, string data [, int flags [, resource context]] )
```

file_put_contents()函数的参数说明如表 12.2 所示。

表 12.2 file_put_contents()函数的参数说明

参　数	说　明
filename	指定写入文件的名称
data	指定写入的数据
flags	实现对文件的锁定。可选值为：FILE_USE_INCLUDE_PATH、FILE_APPEND 或 LOCK_EX，这里只要知道 LOCK_EX 的含义即可，LOCK_EX 为独占锁定
context	一个 context 资源

指点迷津：

本函数可安全用于二进制对象。如果“fopen wrappers”已经被激活，则在本函数中可以把 URL 作为文件名来使用。

例 12.6 通过 fwrite()和 file_put_contents()函数执行文件的写入操作。（**实例位置：配套资源\mr\12\example\12.6**）

首先，应用 file_put_contents()函数写入文件，并应用 file_get_contents()函数读取 bg_01.jpg 图片文件。然后，将读取到的二进制数据通过 file_put_contents()函数写入到另外一个 files.jpg 文件中。最后，通过 img 标记输出 files.jpg 图片。代码如下：

```
<?php
    $path="images/work_03.gif";                    //图片地址和名称
```



```
        $pic=file_get_contents($path);                    //获取数据信息并保存到变量中
        file_put_contents("images/work_04.gif",$pic);     //将图片信息写入到另一张图片中
        echo "<img src='images/work_04.gif'>";            //显示图片
    ?>
```

第二个应用 fwrite()函数完成文件的写入操作。代码如下：

```
    <?php
        $path="images/work_03.gif";                       //图片地址和名称
        $pic=file_get_contents($path);                    //读取图片数据
        $open=fopen("images/work_04.gif","wb");           //以读写二进制方式打开文件
        fwrite($open,$pic);                               //写入信息
        echo "<img src='images/work_04.gif'>";            //输出图像
        fclose($open);                                    //关闭文件
    ?>
```

运行结果如图 12.6 所示。

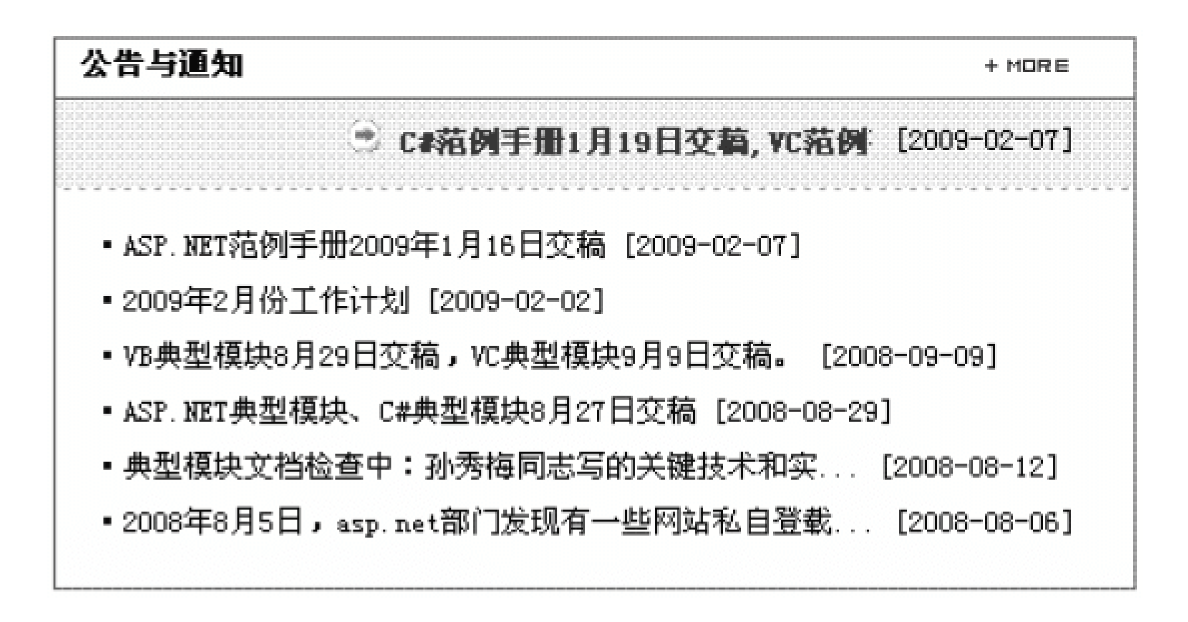


图 12.6　输出写入的二进制文件

12.1.4　关闭文件指针

文件有打开就应该有关闭，对文件的操作结束后，应该关闭这个文件，否则可能会引起错误。在 PHP 中使用 fclose()函数关闭文件。其语法如下：

```
    bool fclose ( resource handle ) ;
```

fclose()函数将参数 handle 指向的文件关闭，成功返回 True，否则返回 False。其中参数 handle（文件指针）必须是有效值，并且是通过 fopen()函数成功打开的文件。有关该函数的应用可以参考上面的实例，这里不再赘述。

上机演练

上机演练 1　文件操作汇总

为了便于对网站的管理、维护和更新，应设计一个能够对文件进行操作的模块，实现对文件的创建、复制、移动和删除等操作，这样能够给网站的管理工作提供很大的方便，不再因为要修改某个文件而要登录到 FTP 中，通过下载或上传实现文件的更新，这种方案能够节省很多的时间。应用文件系统函数实现文件的创建、复制、移动和删除等操作。只要在文本框中输入要复制文件的路径和名称，在对应的文本框中输入指定文件要复制到的具体文件夹的路径和名称（包括指定文件的名称），然后单击“提交”按钮，即可完成文件的复制操作，其运行结果如图 12.7 所示。

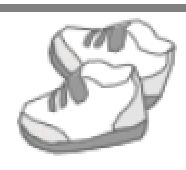
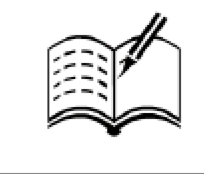




图 12.7　文件操作汇总



上机演练 2　文件类型检测

在开发程序的过程中，很多时候都需要获取到文件的后缀名，并根据后缀名作出一些判断。那么如何才能获取到文件的后缀名称呢？答案是应用$_FILES 全局数组获取上传文件后缀，并根据后缀名称，作出相应的判断。其运行结果如图 12.8 所示。

上机演练 3　删除指定目录下的所有 ini 文件

既然可以对指定目录下的文件进行复制、移动或删除等操作，那么这里将完成一个更加具体的操作，删除指定目录下的所有 ini 文件。其运行结果如图 12.9 所示。



图 12.8　上传文件类型检测



图 12.9　删除指定目录下的所有 ini 文件

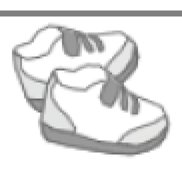
12.2　目录操作技术

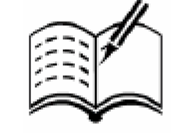
视频讲解：配套资源\mr\12\video\目录操作技术.exe

目录也是文件，严格来说，是一种特殊的文件。那么既然是文件，如果要对其进行操作同样必须要先打开，然后才可以进行浏览、操作，最后还要将其关闭。对于 PHP 目录处理技术的学习仍然可以依照学习文件操作技术的步骤进行。

12.2.1　打开指定目录

打开文件和打开目录虽然都是执行打开的操作，但是使用的函数不同，对未找到指定文件的处理结果也不同。fopen()函数如果未找到指定的文件，那么可能会自动创建这个文件，而打开目录函数 opendir()则会直接抛出一个错误信息。这就是 PHP 提供的打开目录的函数 opendir()。





opendir()函数打开一个指定目录。成功则返回目录句柄，否则返回 False。其语法如下：

```
resource opendir ( string path [, resource context] )
```

参数 path 指定要打开的目录路径，如果参数 path 指定的不是一个有效的目录，或者因为权限、文件系统错误而不能打开等，opendir()函数将返回 False，并产生一个 E_WARNING 级别的错误信息。

指点迷津：

通过在 opendir()函数前添加@符号，可以屏蔽错误信息的输出。

例 12.7 首先验证指定目录是否存在，如果存在则通过 opendir()函数打开指定的目录。其代码如下：（**实例位置：配套资源\mr\12\example\12.7**）

```
<?php
    header("Content-Type:text/html;charset=utf-8");
    if(is_dir("dir")){                          //判断指定文件夹是否存在
        echo "<b style='font-family:楷体_gb2312'>指定文件夹存在</b>";
        echo"<br>";
        $array=scandir('dir') ;                 //读取目录结构
        foreach($array as $key=>$value){
            echo "#".$value."<br>";
        }
    }else{
        echo "<b style='font-family:\'楷体_gb2312\''>指定文件夹不存在</b>";
    }
?>
```

运行结果如图 12.10 所示。



图 12.10　判断指定目录是否存在

12.2.2　读取目录结构

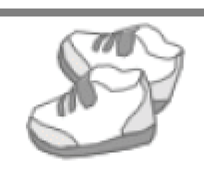
应用 opendir()函数打开目录之后，就可以利用其返回的目录句柄，配合 PHP 中提供的 scandir()函数完成对目录的浏览操作。

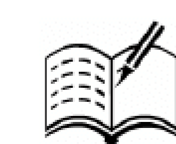
scandir()函数浏览指定路径下的目录和文件。成功返回包含有文件名的 array，失败则返回 False。其语法如下：

```
array scandir ( string directory [, int sorting_order [, resource context]] )
```

参数 directory 指定要浏览的目录，如果 directory 不是目录，那么 scandir()函数将返回 False，并生成一条 E_WARNING 级的错误信息。

参数 sorting_order 设置排序顺序，默认按字母升序排序，如果应用参数 sorting_order，则





变为降序排序。

例 12.8　打开 Apache 服务器的根目录，并且浏览目录下的文件和文件夹，其代码如下：**（实例位置：配套资源\mr\12\example\12.8）**

```
<?php
    $path="../../../../";                                    //定义相对路径
    echo "Apache根目录所在的硬盘路径为：".realpath($path)."<br>";//输出Apache根目录的绝对路径
    if(is_dir($path)){                                       //判断当前路径是否为目录
        $path=scandir($path);                                //将目录信息保存在数组中
        for($a=0;$a<count($path);$a++){                      //循环输出结果
            echo "#".$path[$a]."<br>";
        }
    }
?>
```

运行结果如图 12.11 所示。

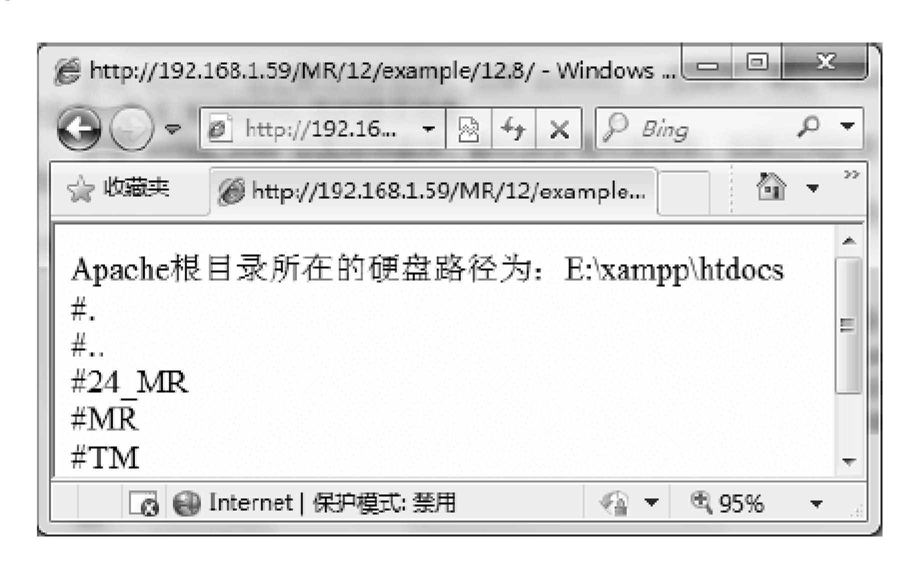


图 12.11　浏览 Apache 服务器的根目录

12.2.3　关闭目录指针

做事情应该有始有终，编程也应该如此。目录打开，完成操作之后，就应该关闭目录。PHP 中通过 closedir()函数关闭目录。其语法如下：

```
void closedir ( resource handle )
```

参数 handle 为使用 opendir()函数打开的一个目录句柄。

在应用 rmdir()函数删除指定的目录时，被删除的路径必须是空的目录，并且权限必须要合乎要求，否则将返回 False。

上机演练

上机演练 4　重新定义目录名称

在对网站进行管理和维护的过程中，经常会修改文件夹的名称，它也是目录的一项基本操作。单击当前目录中文件夹后的“重命名”超链接，将进入到如图 12.12 所示的页面，在该页面中完成对指定目录的重命名。

上机演练 5　获取磁盘分区的大小

通过文件系统函数不但可以对目录、文件进行操作，获取目录文件的相关信息，而且可以获取到磁盘分区的大小。运行本实例，将根据文本框提交的目录，获取到该目录所在磁盘分区



的大小以及该目录下的所有文件。其运行结果如图 12.13 所示。

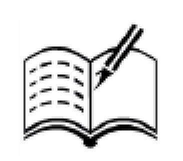






图 12.12　执行目录的重命名

图 12.13　获取磁盘分区的大小

上机演练 6　遍历指定目录下的所有文件

在网站的后台管理系统中，经常需要对网站服务器中的文件进行管理和维护，有时需要添加一个文件夹、删除某个文件夹或文件。为了可以更好地查看到这些文件或文件夹，最好的方法就是创建一个文件查询系统，通过它可以查看到指定文件夹下的所有文件。在文本框中输入一个指定的文件夹，单击“提交”按钮，如果该文件夹存在，就可以显示出该文件夹下包括的所有文件，运行结果如图 12.14 所示。



图 12.14　遍历指定目录下的所有文件

本 章 摘 要

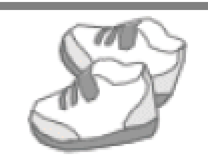
1．讲述文件的基本操作，包括打开、读取、写入和关闭文件，另外在实例中还讲解了文件的判断、修改和删除等操作的方法。

2．讲述目录的基本操作，包括打开、读取，写入和关闭目录，同时还对目录的创建、判断、修改和删除方法进行讲解。

习　　题

1．判断某目录是否存在需要使用（　　）函数。

A．is_float　　B．is_int　　C．is_dir



2．读取目录结构的函数为（　　）。

A．opendir　　B．closedir　　C．scandir　　D．file

3．下面哪个变量可以获取文件的临时名称？（　　）

A．$_FILES　　B．$_POST　　C．$_GET



Note

4．开启文件上传需要修改哪个文件？（　　）

A．phpinfo　　B．php.ini　　C．php.txt

5．创建指定文件目录需要使用函数为（　　）。

A．open　　B．is_dir　　C．mkdir

6．实现将指定文件(name='file')上传并将文件相关信息保存在数组$array 中。请将代码补充完整。

```
if(isset($_POST['sub'])){
  (                                        )
}
```

7．将上传文件移动到指定文件夹需要使用函数（　　　　）。

8．删除目录的函数是（　　　　）。

9．判断指定的文件是否是通过 HTTP POST 上传的需要使用函数（　　）。

10．本部分代码实现的功能是（　　）。

```
header('Content-Type:image/jpg');
header('Content-Disposition:attachment;filename="test.jpg"');
header('Content-Length:'.filesize('test.jpg'));
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时用户可以参考《PHP 经典编程 265 例》一书的第 11 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　可以屏蔽刷新功能的文本计数器

网站的计数器对于网站管理者来说是一个非常值得关注的部分，它记录了网站被访问的次数，客观地反映了网站受欢迎的程度。而文本计数器是最简单的一种，采用将数据存储于文本文件中。运行结果如图 12.15 所示。

最新动态　·　此时是：2010年7月6日16:33:51　您是本网站第1800008 位访客！

图 12.15　可以屏蔽刷新功能的文本计数器

实战模拟 2　从文本文件中读取注册服务条款

在网站开发的过程中，经常会创建注册服务条款或者会员须知之类的文件。处理此类文件的最直接方法是将它生成一个独立的页面，而最实用的方法是将它存储于独立的文本文件中，从文本文件中读取这些服务条款，它不占用数据库的空间，而且不占用过多的页面。其运行结果如图 12.16 所示。









图 12.16　从文本文件中读取注册服务条款

实战模拟 3　遍历、删除指定目录下的所有文件

通过对网站目录的遍历，能够更快地了解网站的结构，网站文件的存储位置；通过对文件内容的遍历，掌握网站中每个文件的作用；同时也实现对目录和文件的管理。其运行结果如图 12.17 所示。



图 12.17　遍历、操作指定目录下的所有文件

第13章

面向对象编程

（自学视频、源程序：配套资源\mr\13\）

面向对象（OOP）的编程方式是PHP的突出特点之一，采用这种编程方式可以对大量零散代码进行有效组织，从而使PHP具备大型Web项目开发的能力。采用面向对象编程方式还可以提高网站的易维护性和易读性。总之，面向对象的编程方式是计算机发展史上具有划时代意义的标志。习惯了面向过程编程思想的程序员，在刚接触面向对象的编程方式时，可能会感觉困难，毕竟二者在开发程序时，无论是方式还是方法都是截然不同的。那么，如何去应对呢？答案是多学习、勤思考，在学习面向对象的编程方式过程中，还应该多读已成型的、优秀的程序，从而全面提高个人的编程能力。以全面掌握面向对象的编程思想为出发点，兼顾面向过程和面向对象两种编程思想为目的，本章将采用讲解与实例相结合的方式、由浅入深，最终使读者能够将面向对象的编程思想应用到实际项目开发中去。

学习摘要：

- 一切皆是对象
- 类的声明，成员方法、成员属性的定义
- 类的实例化
- 访问类中成员
- 构造方法和析构方法
- 面向对象的封装特性
- 面向对象的继承特性
- 抽象类和接口
- 面向对象的多态性
- 面向对象的关键字
- 面向对象的魔术方法





13.1 一切皆是对象

面向对象就是将要处理的问题抽象为对象，然后通过对象的属性和行为来解决对象的实际问题。面向对象的基本概念就是类和对象，接下来将分别进行讲解。

13.1.1 什么是类

世间万物都具有其自身的属性和方法，通过这些属性和方法可以将不同物质区分开来。例如，人具有性别、体重和肤色等属性，还可以进行吃饭、睡觉、学习等活动，这些活动可以说是人具有的功能。可以把人看作程序中的一个类，那么人的性别可以比作类中的属性，吃饭可以比作类中的方法。

也就是说，类是属性和方法的集合，是面向对象编程方式的核心和基础，通过类可以将零散的用于实现某项功能的代码进行有效管理。例如，创建一个数据库连接类，包括 6 个属性：数据库类型、服务器、用户名、密码、数据库和错误处理；包括 3 个方法：成员变量赋值方法、连接数据库方法和关闭数据库方法。数据库连接类的结构如图 13.1 所示。

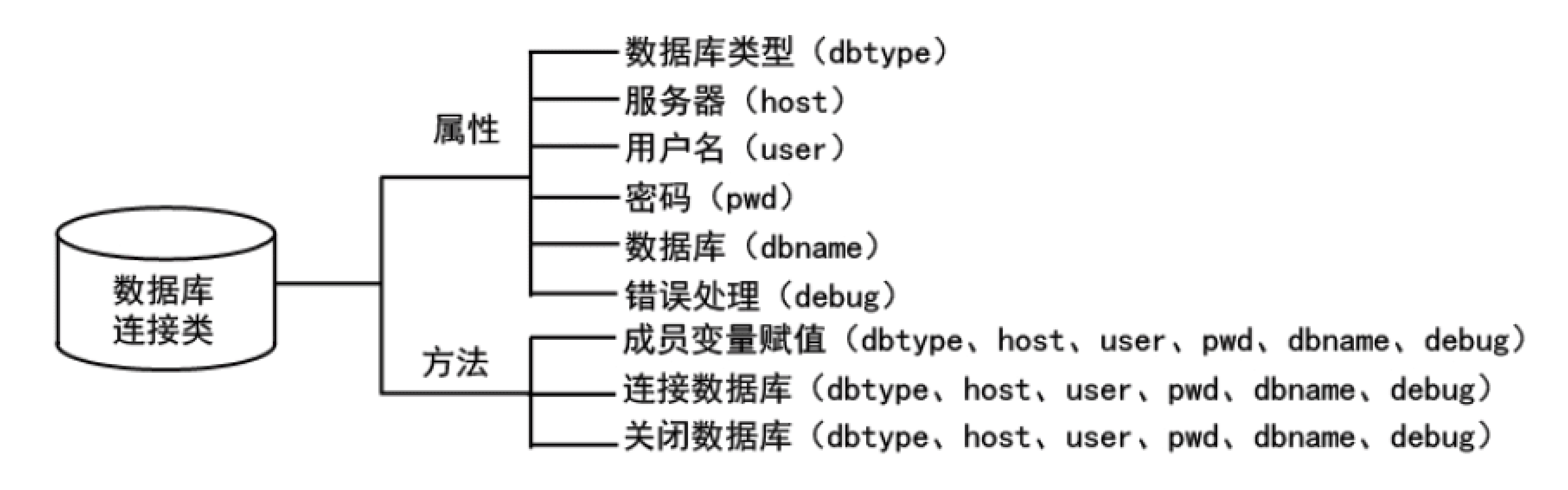


图 13.1 数据库连接类

13.1.2 对象的由来

类只是具备某项功能的抽象模型，在实际应用中还需要对类进行实例化，这样就引入了对象的概念。对象是类进行实例化后的产物，是一个实体。仍然以人为例，“黄种人是人”这句话没有错误，但反过来说“人是黄种人”这句话一定是错误的。因为除了有黄种人，还有黑种人、白种人等。那么“黄种人”就是“人”这个类的一个实例对象。可以这样理解对象和类的关系：对象实际上就是“有血有肉的、能摸得到看得到的”一个类。

这里实例化创建的数据库连接类，调用数据库连接类中的方法，并完成与数据库的连接操作，如图 13.2 所示。

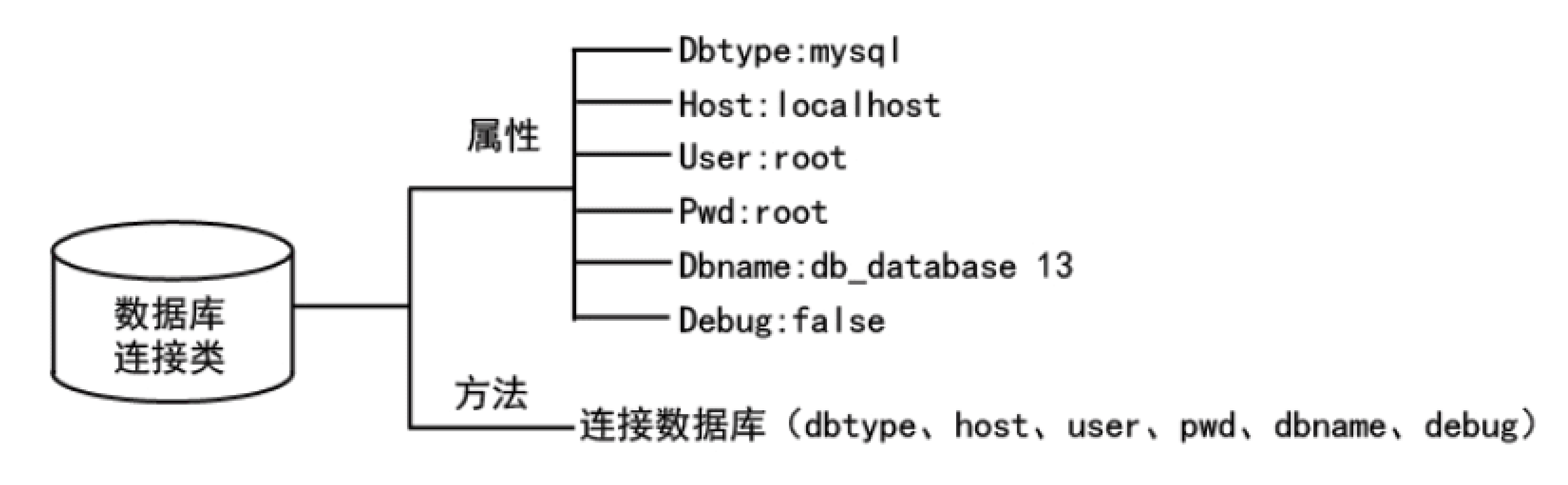
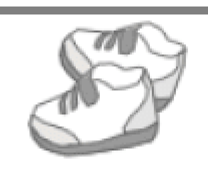


图 13.2 实例化对象





13.1.3　面向对象的特点

面向对象编程的 3 个重要特点是：继承、封装和多态，它们迎合了编程中注重代码重用性、灵活性和可扩展性的需要，奠定了面向对象在编程中的地位。

（1）封装性：就是将一个类的使用和实现分开，只保留有限的接口（方法）与外部联系。对于使用该类的开发人员，只要知道该类该如何使用即可，而不用去关心该类是如何实现的。这样做可以让开发人员更好地把精力集中起来专注其他事情，同时也避免了程序之间的相互依赖而带来的不便。

例如，使用电脑时，并不需要将电脑拆开了解每个部件的具体用处，用户只需按下主机箱上的 Power 按钮就可以启动电脑。但对于电脑内部的构造，用户可以不必了解，这就是封装的具体表现。

（2）继承性：是派生类（子类）自动继承一个或多个基类（父类）中的属性与方法，并可以重写或添加新的属性或方法。继承这个特性简化了对象和类的创建，增加了代码的可重用性。

假如已经定义了 A 类，接下来准备定义 B 类，而 B 类中有很多属性和方法与 A 类相同，那么就可以使 B 类继承于 A 类，这样就无须再在 B 类中定义 A 类已有的属性和方法，从而可以在很大程度上提高程序的开发效率。

例如，定义一个水果类，水果类具有颜色属性，然后定义一个苹果类，在定义苹果类时完全可以不定义苹果类的颜色属性，通过如图 13.3 所示的继承关系完全可以使苹果类具有颜色属性。

（3）多态性：指同一个类的不同对象，使用同一个方法可以获得不同的结果。多态性增强了软件的灵活性和重用性。

例如，定义一个火车类和一个汽车类，火车和汽车都可以移动，说明两者在这方面可以进行相同的操作，然而，火车和汽车移动的行为是截然不同的，因为火车必须在铁轨上行驶，而汽车在公路上行驶，这就是类多态性的形象比喻，如图 13.4 所示。

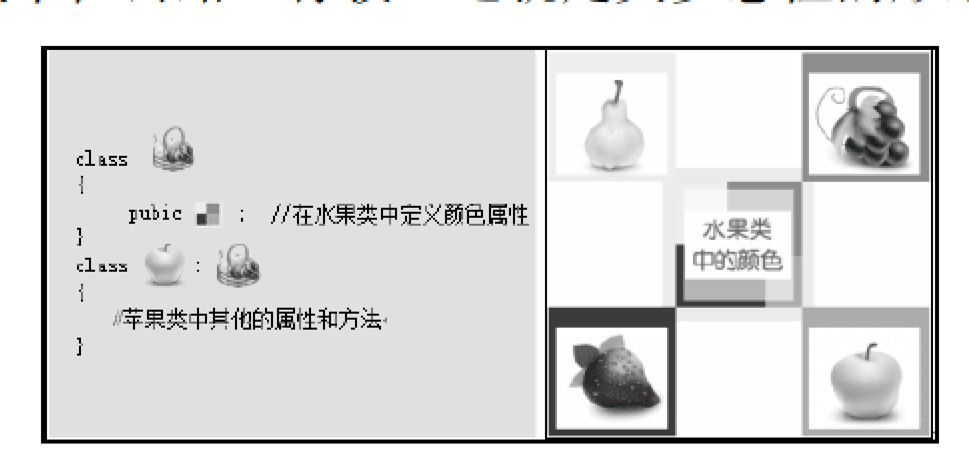


图 13.3　继承特性效果示意图



图 13.4　多态在生活中的体现

13.2　类 的 声 明

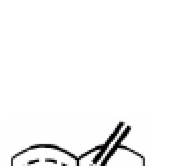
视频讲解：配套资源\mr\13\video\类的声明.exe

在创建类名称时必须将类进行声明。

13.2.1　类的定义

和很多面向对象的语言一样，PHP 也是通过 class 关键字加类名来定义类的。类的格式如下：





Note

```
<?php
    权限修饰符    class    类名{
        类体;
    }
?>
```

☑ 权限修饰符是可选项，可以使用 public、protected、private 或省略这 3 者。

☑ class 是创建类的关键字。

☑ 类名是所要创建类的名称，必须写在 class 关键字之后，在类的名称后面必须跟上一对大括号。

☑ 类体是类的成员，类体必须放在类名后面的两个大括号“{”和“}”之间。

说明：

在创建类时，在 class 关键字前除了可以加权限修饰符外，还可以加其他关键字如 static、abstract 等，有关创建类使用的权限修饰符和其他关键字将在后面的内容中进行讲解。至于类名的定义，与变量名和函数名的命名规则类似，如果由多个单词组成，则习惯上每个单词的首字母要大写，并且类名应该有一定的意义。

例如，创建一个 ConnDB 类。代码如下：

```
<?php
class ConnDB{            //定义数据库连接类
    //…
}
?>
```

多学两招：

虽然 ConnDB 类仅有一个类的骨架，任何功能都没有实现，但这并不影响它的存在。一个类即一对大括号之间的全部内容都要在一段代码段中，不允许将类中的内容分割成块，例如：

```
<?php
class ConnDB{            //定义数据库连接类
    //…
?>
<?php
    //…
}
?>
```

这种格式是不允许的。

13.2.2 成员属性

在类中直接声明的变量称为成员属性（也可以称为成员变量），可以在类中声明多个变量，即对象中有多个成员属性，每个变量都存储对象不同的属性信息。

成员属性的类型可以是 PHP 中的标量类型和复合类型，但是如果使用资源和空类型则是没有意义的。





成员属性的声明必须有关键字来修饰，例如 public、protected、private 等，这是一些具有特定意义的关键字。如果不需要有特定的意义，那么可以使用 var 关键字来修饰。另外，在声明成员属性时没有必要赋初始值。

下面再次创建 ConnDB 类并在类中声明一些成员属性，其代码如下：

```
class ConnDB{                                   //定义类
    var  $dbtype;                               //声明成员属性
    var  $host;                                 //声明成员属性
    var  $user;                                 //声明成员属性
    var  $pwd;                                  //声明成员属性
    var  $dbname;                               //声明成员属性
    var  $debug;                                //声明成员属性
    var  $conn;                                 //声明成员属性
}
```

13.2.3 成员方法

在类中声明的函数称为成员方法。一个类中可以声明多个函数，即对象中可以有多个成员方法。成员方法的声明和函数的声明是相同的，唯一特殊之处是成员方法可以有关键字来对它进行修饰，控制成员方法的权限。声明成员方法的代码如下：

```
class ConnDB{                                   //定义类
    function ConnDB(){                          //声明构造方法
    //方法体
 }
 function GetConnId(){                          //声明数据库连接方法
    //方法体
 }
 function CloseConnId(){                        //声明数据库关闭方法
        $this->conn->Disconnect();              //方法体，执行关闭的操作
    }
}
```

在类中成员属性和成员方法的声明都是可选的，可以同时存在，也可以单独存在，具体应根据实际的需求而定。

上机演练

上机演练 1　使用类的属性保存数据库连接参数

通过类的属性保存数据库连接参数，如图 13.5 所示。首先在文本框内输入连接 MySQL 数据库服务器的必选参数，然后单击“连接”按钮即可建立与指定的 MySQL 数据库服务器的连接，并将连接结果打印在页面中。

上机演练 2　数据库连接类中定义数据库连接方法

使用 PHP 的 MySQL 函数库管理 MySQL 数据库时，首先需要获得数据库连接句柄，然后才能进一步进行增、删、改、查操作。在数据库连接类中，定义数据库连接方法，实现与数据库的连接，如图 13.6 所示。



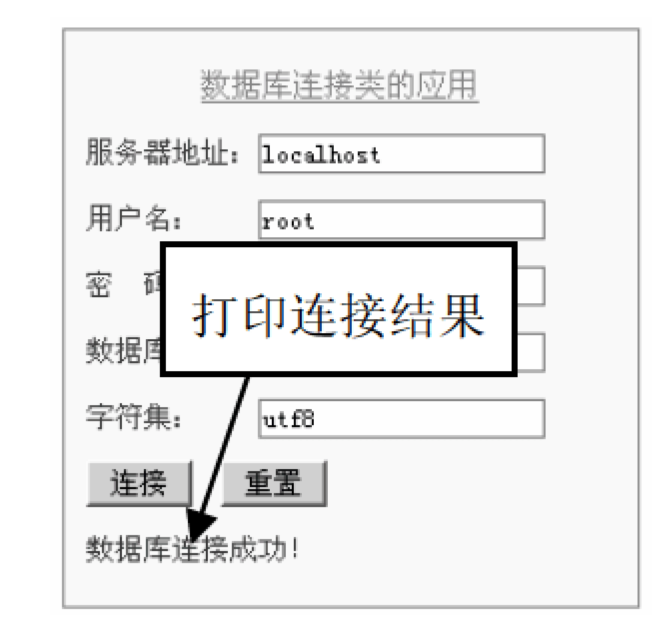




图 13.5　数据库连接成功

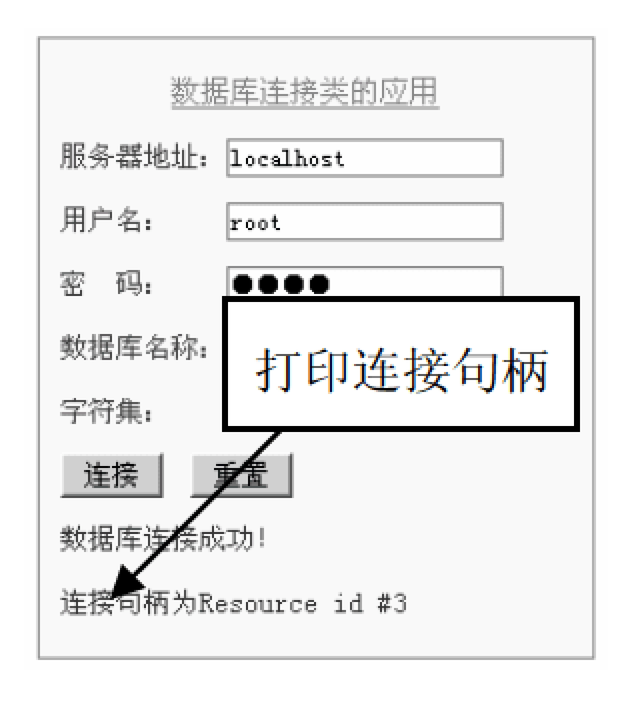


图 13.6　打印连接句柄

上机演练 3　数据统计类中定义求数值平均数的方法

封装数据统计类，定义求数值平均数的方法，如图 13.7 所示。首先在该图的文本框中输入一组数字并用半角逗号分割，然后单击“计算”按钮将在页面中打印出该组数字的平均值。

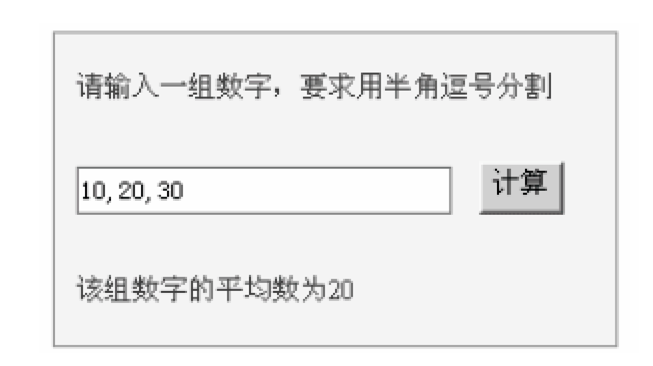


图 13.7　计算一组数的平均值

13.3　类的实例化

视频讲解：配套资源\mr\13\video\类的实例化.exe

13.3.1　创建对象

面向对象程序的最终操作者是对象，而对象是类实例化的产物，所以学习面向对象只停留在类的声明上是不够的，必须学会将类实例化成对象。类的实例化格式如下：

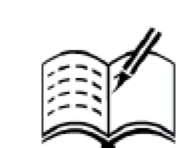
```
$变量名=new 类名称([参数]);                          //类的实例化
```

- ☑ $变量名：类实例化返回的对象名称，用于引用类中的方法。
- ☑ new：关键字，表明要创建一个新的对象。
- ☑ 类名称：表示新对象的类型。
- ☑ 参数：指定类的构造方法用于初始化对象的值。如果类中没有定义构造函数，则 PHP 会自动创建一个不带参数的默认构造函数。

例如，这里对上面创建的 ConnDB 类进行实例化。其代码如下：

```
class ConnDB{                                       //定义类
      function ConnDB(){                            //声明构造方法
      //方法体
 }
 function GetConnId(){                              //声明数据库连接方法
      //方法体
 }
```





```
    function CloseConnId(){                              //声明数据库关闭方法
            $this->conn->Disconnect();                   //方法体，执行关闭的操作
        }
    }
    $connobj1=new ConnDB();                              //类的实例化
    $connobj2=new ConnDB();                              //类的实例化
    $connobj3=new ConnDB();                              //类的实例化
```

一个类可以实例化多个对象，每个对象都是独立的。如果上面的 ConnDB 类实例化了 3 个对象，就相当于在内存中开辟了 3 个空间存放对象。同一个类声明的多个对象之间没有任何联系，只能说明它们是同一个类型，就如同是 3 个人，他们都有自己的姓名、身高、体重，都可以进行吃饭、睡觉、学习等活动。

13.3.2 访问类中成员

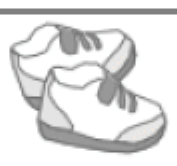
在类中包括成员属性和成员方法，访问类中的成员包括成员属性和方法的访问。访问方法与访问数组中的元素类似，需要通过对象的引用来访问类中的每个成员。其中还要应用一个特殊的运算符号“->”。访问类中成员的语法格式如下：

```
    $变量名=new 类名称([参数]);                           //类的实例化
    $变量名->成员属性=值;                                 //为成员属性赋值
    $变量名->成员属性;                                    //直接获取成员属性值
    $变量名->成员方法;                                    //访问对象中指定的方法
```

这是访问类中成员的基本格式，下面看它们在具体的实例中是如何运用的。

例 13.1 创建 ConnDB 类，对类进行实例化，并访问类中的成员属性和成员方法。代码如下：**（实例位置：配套资源\mr\13\example\13.1）**

```
    <?php
     class mysql{                                        //定义数据库连接类
        var $localhost;                                  //定义成员变量
        var $name;
        var $pwd;
        var $db;
        var $conn;
        public function mysql($localhost,$name,$pwd,$db){ //定义构造方法
            $this->localhost=$localhost;                 //为成员变量赋值
            $this->name=$name;
            $this->pwd=$pwd;
            $this->db=$db;
            $this->connect();
        }
        public function connect(){                       //定义数据库连接方法
            $this->conn=$conn;
            $this->conn=mysql_connect($this->localhost,$this->name,$this->pwd)or die("CONNECT
MYSQL FALSE");                                           //执行连接操作
            mysql_select_db($this->db,$this->conn)or die("CONNECT DB FALSE");    //选择数据库
            mysql_query("SET NAMES utf8");               //设置数据库编码格式
        }
```




```
        public function GetId(){                         //定义方法，返回数据库连接信息
            echo "MYSQL服务器的用户名：".$this->name."<br>";
            echo "MYSQL服务器的密码：".$this->pwd;
        }
    }
    $msl=new mysql("127.0.0.1","root","111","db_database13");   //实例化数据库连接类
    $msl->GetId();                                             //调用类中方法
?>
```

运行结果如图 13.8 所示。



图 13.8　访问类中成员的结果

13.3.3　特殊的访问方法——“$this”和“::”

1. $this

在例 13.1 中使用了一个特殊的对象引用方法“$this”，那么它到底表示什么意义呢？在这里将进行详细讲解。

$this 存在于类的每个成员方法中，它是一个特殊的对象引用方法。成员方法属于哪个对象，$this 引用就代表哪个对象，其作用就是专门完成对象内部成员之间的访问。

正如在例 13.1 中定义的那样，将传递的参数值直接赋给成员变量，而在 GetConnId()方法中，直接通过$this->user 和$this->pwd 获取数据库的用户名和密码。

2. 操作符“::”

相比$this 引用只能在类的内部使用，操作符“::”的功能更加强大，其可以在没有声明任何实例的情况下访问类中的成员。例如，在子类的重载方法中调用父类中被覆盖的方法。操作符“::”的语法格式如下：

```
关键字::变量名/常量名/方法名
```

这里的关键字分为 3 种情况：

☑　parent 关键字：可以调用父类中的成员变量、成员方法和常量。

☑　self 关键字：可以调用当前类中的静态成员和常量。

☑　类名：可以调用本类中的变量、常量和方法。

例 13.2　本实例依次使用类名、parent 关键字和 self 关键字来调用变量和方法。读者可以观察输出的结果。代码如下：（**实例位置：配套资源\mr\13\example\13.2**）

```
<?php
/*
    当实例化对象后不需要使用对象句柄调用对应的方法时可以只给类实例化不返回对象句柄
*/
    class Car{
        const NAME="别克系列";
```






```
            public function __construct(){           //定义构造方法
                echo "父类：".Car::NAME;              //类名引用
            }
        }
        class SmallCar extends Car{                  //继承
            const NAME="别克君威";
            public function __construct(){           //定义构造方法
                echo parent::__construct()."\t";     //应用父类构造方法
                echo "子类：".self::NAME;
            }
        }
        new SmallCar();                              //实例化对象
    ?>
```

运行结果如图 13.9 所示。



图 13.9　关键字的使用

13.3.4　构造方法和析构方法

1. 构造方法

构造方法是对象创建完成后第一个被对象自动调用的方法，它存在于每个声明的类中，是一个特殊的成员方法。如果在类中没有直接声明构造方法，那么在类中将会默认生成一个没有任何参数且内容为空的构造方法。

构造方法多数是执行一些初始化的任务。例如，例 13.1 中通过构造方法为成员变量赋初始值。

在 PHP 中，构造方法的声明有两种情况：第一种在 PHP5 以前的版本中，构造方法的名称必须与类名相同；第二种在 PHP5 的版本中，构造方法的方法名称必须是以两个下划线开始的“__construct()”。虽然在 PHP5 中构造方法的声明方法发生了变化，但是以前的方法还是可用的。

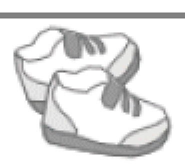
PHP5 中的这个变化是考虑到构造函数可以独立于类名，当类名发生变化时不需要修改相应的构造函数的名称。通过__construct()声明构造方法的语法格式如下：

```
function __construct([mixed args [,…]]){
    //方法体
}
```

在 PHP 中，一个类只能声明一个构造方法。在构造方法中可以使用默认参数，实现其他面向对象的编程语言中构造方法重载的功能。如果在构造方法中没有传入参数，那么将使用默认参数为成员变量进行初始化。

例 13.3　在例 13.1 中，通过使用与类名相同的方法声明构造方法，那么这里将通过__construct()声明一个与类名不同的构造方法。代码如下：（**实例位置：配套资源\mr\13\example\13.3**）

```
<?php
/*
```



Note

```
        构造函数：当类被实例化后构造函数自动执行。所以如果用户希望在实例化的同时调用某
个方法可以把此方法通过this关键字调用
    */
    class mysql{                                                       //定义类名称
        var $localhost;                                                //定义变量
        var $name;
        var $pwd;
        var $db;
        var $conn;
        public function __construct($localhost,$name,$pwd,$db){        //构造函数
            $this->localhost=$localhost;
            $this->name=$name;
            $this->pwd=$pwd;
            $this->db=$db;
            $this->connect();
        }
        //省略了部分代码
    }
    $msl=new mysql("127.0.0.1","root","111","db_database13");          //实例化对象
    $msl->GetId();                                                     //对象句柄调用指定的方法
?>
```

其运行结果与例 13.1 相同。

2. 析构方法

析构方法的作用和构造方法正好相反，是对象被销毁之前最后一个被对象自动调用的方法。它是 PHP5 中新添加的内容，实现在销毁一个对象之前执行一些特定的操作，诸如关闭文件、释放内存等。

析构方法的声明格式与构造方法类似，都是以两个下划线开头的“__destruct”，析构函数没有任何参数。其语法格式如下：

```
function __destruct(){
    //方法体，通常是完成一些在对象销毁前的清理任务
}
```

在 PHP 中有一种“垃圾回收”机制，可以自动清除不再使用的对象，释放内存。而析构方法就是在这个垃圾回收程序执行之前被调用的方法，在 PHP 中它属于类中的可选内容。

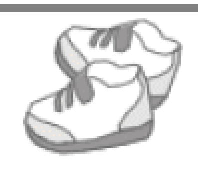
上机演练

上机演练 4　使用$this 关键字调用汽车类自身的方法

通过$this 关键字在汽车类内部调用设置汽车颜色的方法和设置汽车品牌的方法，输出汽车的信息，如图 13.10 所示。分别选择表单中的颜色和类型单选按钮，然后单击“提交”按钮即可在图中打印出汽车信息。

上机演练 5　学生类中使用构造方法为学生信息初始化

采用二维数组模拟数据库存储学生信息，然后通过 for 循环语句实例学生类，并通过构造



方法为学生类的属性赋初值，最后通过类的 getXxx 方法获得学生属性，并将学生信息输出在页面上，如图 13.11 所示。



图 13.10　打印汽车信息

学号	姓名	年龄	住址
0312310	小明	16	北京西城区
0312311	小张	16	北京宣武区
0312312	小赵	17	北京海淀区

图 13.11　学生信息列表

13.4　面向对象的封装特性

视频讲解：配套资源\mr\13\video\面向对象的封装特性.exe

面向对象编程的特点之一是封装性，将类中的成员属性和方法结合成一个独立的相同单位，并尽可能隐藏对象的内容细节。其目的就是确保类以外的部分不能随意存取类的内部数据（成员属性和成员方法），从而有效避免外部错误对类内数据的影响。

类的封装是通过关键字 public、private、protected、static 和 final 来实现的。下面对其中的 public、private 和 protected 关键字进行详细讲解。

13.4.1　public（公共成员）

顾名思义，public（公共成员）就是可以公开的、没有必要隐藏的数据信息，可以在程序的任何地点（类内、类外）被其他的类和对象调用。子类可以继承和使用父类中所有的公共成员。

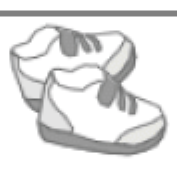
在本节内容的前半部分，所有的变量都被声明为 public，而所有的方法在默认的状态下也是 public，所以对变量和方法的调用显得十分混乱。为了解决这个问题，就需要使用第二个关键字——private。

13.4.2　private（私有成员）

被 private 关键字修饰的变量和方法，只可以在所属类的内部被调用和修改，不可以在类外被访问，即使是子类中也不可以。

例 13.4　通过调用成员方法对私有变量$name 进行修改与访问；如果直接调用私有变量，将会发生错误。代码如下：**（实例位置：配套资源\mr\13\example\13.4）**

```
<?php
    class Car{
        private $carName="奥迪系列";                    //定义私有变量并赋值
        public function setName($carName){          //利用set()方法为设置变量值
            $this->carName=$carName;
        }
        public function getName(){                      //利用get()方法返回变量值
            return $this->carName;
```




```
            }
        }
        class SmallCar extends Car{                         //继承
        }
        $car=new SmallCar();                                //实例化子类对象
        $car->setName("Q7");                                //为子类变量赋值
        echo "正确操作私有变量<br>";
        echo $car->getName();                               //输出子类变量的值
        echo "<br>错误操作私有变量";
        echo Car::$carName;
    ?>
```

运行结果如图 13.12 所示。



图 13.12　private 关键字

多学两招：

对于成员方法，如果没有写关键字，那么默认就是 public。从本节开始，以后所有的方法及变量都会带上关键字，这是一种良好的编程习惯。

13.4.3　protected（保护成员）

private 关键字可以将数据完全隐藏起来，除了在本类外，其他地方都不可以调用，子类也不可以。但对于有些变量希望子类能够调用，但对另外的类来说，还要做到封装。这时，就可以使用 protected 关键字。被 protected 关键字修饰的类成员，可以在本类和子类中被调用，其他地方则不可以被调用。

例 13.5　首先声明一个 protected 变量，然后使用子类中的方法调用，最后在类外直接调用一次。代码如下：（**实例位置：配套资源\mr\13\example\13.5**）

```
    <?php
        class Car{                                          //定义轿车类
            protected $carName="奥迪系列";                  //定义保护变量
        }
        class SmallCar extends Car{                         //小轿车类定义轿车类
            public function say(){                          //定义say方法
                echo "调用父类中的属性：".$carName=$this->carName;     //输出父类变量
            }
        }
        $car=new SmallCar();                                //实例化对象
        $car->say();                                        //调用say方法
```



```
        $car->$carName='奥迪Q7';
?>
```

运行结果如图 13.13 所示。



图 13.13　protected 关键字



多学两招：

虽然 PHP 中没有对修饰变量的关键字做强制性的规定和要求，但从面向对象的特征和设计方面考虑，一般使用 private 或 protected 关键字来修饰变量，以防止变量在类外被直接修改和调用。

上机演练

上机演练 6　汽车类使用 public 关键字定义汽车的行使方法

在类体外、类内部的方法中和类的子类中调用被 public 关键字所修饰的方法，运行结果如图 13.14 所示。

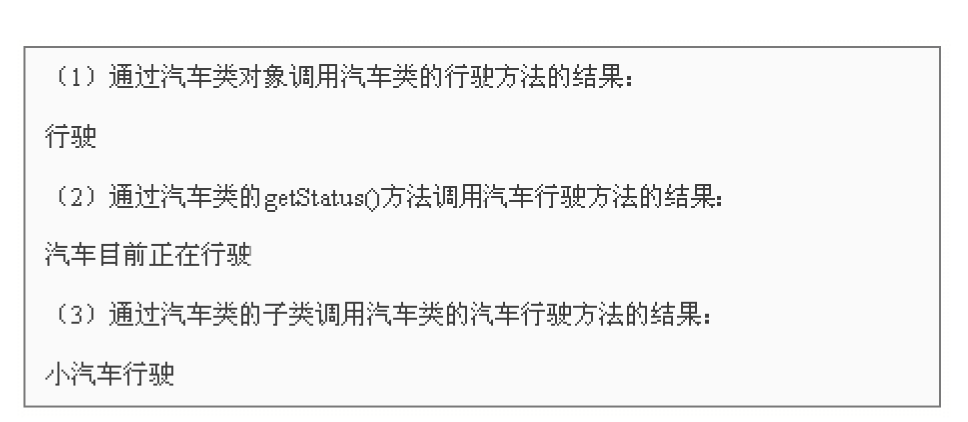


图 13.14　被 public 关键字修饰的方法在不同范围内被调用

上机演练 7　使用 private 关键字定义汽车的颜色属性

通过在不同范围内调用类中私有属性来演示私有成员的作用范围。如果通过类实例后的属性直接调用类的私有属性，将在页面打印如图 13.15 所示的错误信息；如果通过类中方法调用类的私有属性将输出汽车的颜色，如图 13.16 所示。

通过类实例的对象调用类的私有属性的结果：

Fatal error: Cannot access private property Car::$color in **D:\AppServ\www\MR\13\13\paradigm\07\index.php** on line **17**

图 13.15　通过对象调用类中私有属性

在类中的方法中调用类的私有属性的结果：

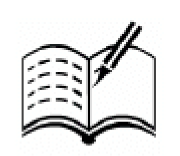
汽车颜色是：红色

图 13.16　通过类中方法调用类的私有属性

上机演练 8　使用 protected 关键字定义汽车的保修年限

使用 protected 关键字定义汽车的保修年限，用于说明类中的保护成员在不同范围内被调用

的结果。当使用类实例的对象调用类中的保护属性时，将出现如图 13.17 所示的错误提示，然后注释掉通过类对象调用类体内保护属性的代码，并分别通过在类体内和类的子类中调用类的保护属性，再次运行将出现如图 13.18 所示的页面。



通过类实例的对象直接调用类中的保护成员的结果

Fatal error: Cannot access protected property Car::$repairTime in **D:\AppServ\www\MR\13\13\paradigm\08\index.php** on line **17**

图 13.17　通过对象调用类中保护成员

在类体内调用保护成员的结果
汽车保修年限为：3年
在类子类中调用保护成员的结果
小汽车保修年限为：5年

图 13.18　通过子类调用父类的保护成员

13.5　面向对象的继承特性

视频讲解：配套资源\mr\13\video\面向对象的继承特性.exe

面向对象编程的特点之二是继承性，使一个类继承并拥有另一个已存在类的成员属性和成员方法，其中被继承的类称为父类，继承的类称为子类。通过继承能够提高代码的重用性和可维护性。

13.5.1　类的继承——extends 关键字

类的继承是类与类之间的一种关系的体现。子类不仅有自己的属性和方法，而且还拥有父类的所有属性和方法。

在 PHP 中，类的继承通过 extends 关键字实现，其语法格式如下：

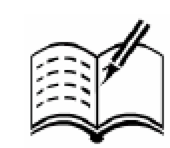
```
class 子类名称 extends 父类名称{
    //子类成员变量列表
    function 成员方法(){                  //子类成员方法
        //方法体
    }
    //省略其他方法
}
```

读者应该记得在 13.1.3 节中通过一个水果父类和一个苹果子类来形象地比喻面向对象继承性的特点。下面就创建这个子类和父类，体会一下它们之间的继承关系。

例 13.6　创建一个水果父类，然后在另一个苹果类中通过 extends 关键字来继承水果类中的成员属性和方法，最后对子类进行实例化操作。代码如下。（**实例位置：配套资源\mr\13\example\13.6**）

```
<?php
    class Fruit{
        var $apple="苹果";                //定义变量
        var $banana="香蕉";
        var $orange="橘子";
    }
    class FruitType extends Fruit{        //类之间继承
        var $grape="葡萄";                //定义子类变量
```





```
        }
        $fruit=new FruitType();                //实例化对象
        echo "水果包含：".$fruit->apple."，".$fruit->banana."，".$fruit->orange."，".$fruit->grape;
?>
```

运行结果如图 13.19 所示。



图 13.19　类的继承

13.5.2　类的继承——parent::关键字

通过 parent::关键字也可以在子类中调用父类中的成员方法，其语法格式如下：

```
parents::父类的成员方法(参数);
```

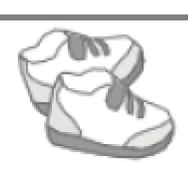
下面通过 parent::关键字重新设计例 13.6 中的继承方法。在子类的 AppleFruit_Type 方法中，直接通过 parent::关键字调用父类中的 FruitType 方法。其关键代码如下：

```
<?php
    class Fruit{                              //定义水果类
        var $apple="苹果";                    //定义变量
        var $banana="香蕉";
        var $orange="橘子";
        public function say(){                //定义say方法
            echo "，".$this->apple."，";       //利用this关键字输出本类中的变量
            echo $this->banana."，";
            echo $this->orange;
        }
    }
    class FruitType extends Fruit{            //类之间继承
        var $grape="葡萄";                    //定义子类变量
        public function show(){               //定义show方法
            parent::say();                    //利用关键字parent调用父类中的say方法
        }
    }
    $fruit=new FruitType();                   //实例化对象
    echo $fruit->grape;                       //调用子类变量
    $fruit->show();                           //调用子类show方法
?>
```

输出结果与例 13.6 相同。

13.5.3　覆盖父类方法

所谓覆盖父类方法，就是使用子类中的方法将从父类中继承的方法进行替换，也叫方法的重写。覆盖父类方法的关键就是在子类中创建与父类中相同的方法，包括方法名称、参数和返回值类型。



例如，可以在子类中创建一个与父类方法同名的方法，那么就实现了方法的重写。其关键代码如下：

```
<?php
    class Car{                                  //定义轿车类
        protected $wheel;                       //定义保护变量
        protected $steer;
        protected $speed;
        public function say_type(){             //定义轿车类型方法
            $this->wheel="45.9cm";              //定义车轮直径长度
            $this->steer="15.7cm";              //定义方向盘直接长度
            $this->speed="120m/s";              //定义车速
        }
    }
    class SmallCar extends Car{                 //定义小型轿车类继承轿车类
        public function say_type_Q7(){          //定义Q7轿车类型
            $this->wheel="50.9cm";              //定义车轮直径长度
            $this->steer="20cm";                //定义方向盘直径长度
            $this->speed="160m/s";              //定义车速
        }
        public function say_show(){             //定义输出方法
            $this->say_type_Q7();               //调用本类中方法
            echo "Q7轿车轮胎尺寸：".$this->wheel."<br>"; //输出本类中定义的车轮直径长度
            echo "Q7轿车方向盘尺寸：".$this->steer."<br>"; //输出本类中定义方向盘直径长度
            echo "Q7轿车最高时速：".$this->speed;          //输出本类中定义的最高时速
        }
    }
    $car=new SmallCar();                                   //实例化小轿车类
    $car->say_show();                                      //调用say_show方法
?>
```

运行效果如图 13.20 所示。



图 13.20　重写方法

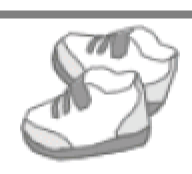
提示：

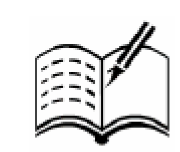
如果父类和子类中都定义了构造方法，当子类的对象被创建后，将调用子类的构造方法，而不会调用父类的构造方法。

上机演练

上机演练 9　苹果子类继承水果父类

首先定义父类水果类，并在该类中定义获得水果颜色和水果形状的方法，然后定义水果类





的子类苹果类，在苹果类中只定义构造方法用于实现对类进行初始化，然后通过调用父类的方法获得苹果的颜色和形状。如图 13.21 所示，首先在表单中输入苹果的颜色和形状属性，然后单击“提交”按钮即可在页面中打印出苹果的颜色和形状属性。

上机演练 10　苹果子类中覆盖水果父类中的方法

应用类的重写机制，在苹果类中重写其父类水果类的 getColor 方法，运行结果如图 13.22 所示。其中第一行为通过水果类调用其自身的 getColor 方法的输出结果，第二行为苹果类调用其自身的 getColor 方法的输出结果，并且该方法重写了其父类水果类的 getColor 方法。

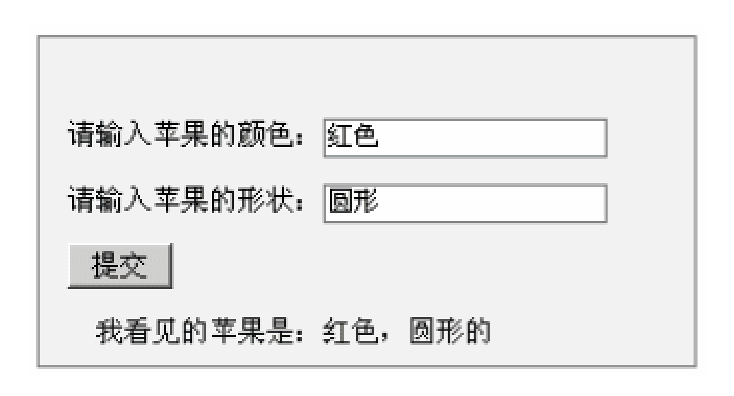


图 13.21　打印苹果的颜色和形状属性

不同的水果颜色不同，无法确定

苹果的颜色是红色

图 13.22　重写水果类中的 getColor()方法的结果

13.6　抽象类和接口

视频讲解：配套资源\mr\13\video\抽象类和接口.exe

抽象类（Abstract）和接口（Interface）都是不能被实例化的特殊类，同时它们都是配合面向对象的多态性一起使用。下面讲解它们的声明和使用方法。

13.6.1　抽象类

抽象类是一种不能被实例化的类，只能作为其他类的父类来使用。抽象类使用 abstract 关键字来声明，其语法格式如下：

```
abstract class 抽象类名称{
    //抽象类的成员变量列表
    abstract function 成员方法1( 参数 );        //定义抽象方法
    abstract function 成员方法2( 参数 );        //定义成员方法
}
```

抽象类和普通类相似，包含成员变量和成员方法。两者的区别在于，抽象类至少要包含一个抽象方法。抽象方法没有方法体，其功能的实现只能在子类中完成。另外，抽象方法也是使用 abstract 关键字来修饰。

注意：

在抽象方法后面要有分号“;”。

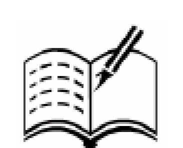
抽象类和抽象方法主要应用于复杂的层次关系中，这种层次关系要求每一个子类都包含并重写某些特定的方法。

例如，中国的美食是多种多样的，有吉菜、鲁菜、川菜、粤菜等。每种菜系使用的都是煎、炒、烹、炸等手法，只是在具体的步骤上，各有各的不同。如果把中国美食当作一个大类 Cate，下面的各大菜系就是 Cate 的子类，而煎炒烹炸则是每个类中都有的方法。每个方法在子类中的实现都是不同的，在父类中无法规定。为了统一规范，不同子类的方法要有一个相同的方法



名：decoct（煎）、stir_fry（炒）、cook（烹）、fry（炸）。

例 13.7 根据中国的美食，首先，创建一个抽象类 Cate，在抽象类中定义 4 个抽象方法：decocts（煎）、stir_frys（炒）、cooks（烹）、frys（炸）。接着，创建吉、鲁、川、粤 4 个菜系子类，继承 Cate 类，并在子类中定义抽象方法：decocts（煎）、stir_frys（炒）、cooks（烹）、frys（炸）。最后，实例化吉菜子类。关键代码如下：**（实例位置：配套资源\mr\13\example\13.7）**



Note

```
<?php
    abstract class cate{                              //定义抽象类
        abstract function decocts();                  //定义抽象方法煎
        abstract function stir_frys();                //定义抽象方法炒
        abstract function cooks();                    //定义抽象方法烹
        abstract function frys();                     //定义抽象方法炸
    }
    class JL_Cate{                                    //定义吉菜
        public function decocts($a,$b){               //定义煎方法
            echo "您点的菜是：".$a."<br>";            //输出菜名
            echo "价格是：".$b."<br>";                //输出价格
        }
        public function stir_frys($a,$b){             //定义炒方法
            echo "您点的菜是：".$a."<br>";            //输出菜名
            echo "价格是：".$b."<br>";                //输出价格
        }
        public function cooks($a,$b){                 //定义烹方法
            echo "您点的菜是：".$a."<br>";            //输出菜名
            echo "价格是：".$b."<br>";                //输出价格
        }
        public function frys($a,$b){                  //定义炸方法
            echo "您点的菜是：".$a."<br>";            //输出菜名
            echo "价格是：".$b."<br>";                //输出价格
        }
    }
//省略了部分代码
    $jl=new JL_Cate();                                //实例化吉菜系
    $jl->decocts("小鸡炖粉条","39元");                //调用煎方法
?>
```

运行结果如图 13.23 所示。



图 13.23　抽象类的应用

13.6.2　接口

继承特性简化了对象、类的创建，增加了代码的可重性，但 PHP 只支持单继承。如果想



实现多重继承，就要使用接口，PHP 可以实现多个接口。

1. 接口的声明

接口类通过 interface 关键字来声明，接口中声明的方法必须是抽象方法，且不能声明变量，只能使用 const 关键字声明为常量的成员属性，另外，接口中的所有成员都必须具备 public 的访问权限。接口声明的语法格式如下：

```
interface 接口名称{                      //使用interface关键字声明接口
    常量成员                             //接口中成员只能是常量
    抽象方法;                            //成员方法必须是抽象方法
}
```

接口和抽象类相同，都不能进行实例化的操作，也都需要通过子类来实现。但不同的是，接口可以直接使用接口名称在接口外去获取常量成员的值。

例如，下面声明一个 One 接口，其代码如下：

```
interface One{                                  //声明接口
    const CONSTANT='CONSTANT value';            //声明常量成员属性
    function FunOne();                          //声明抽象方法
}
```

接口之间也可以实现继承，同样需要使用 extends 关键字。

例如，下面声明一个 Two 接口，通过 extends 关键字继承 One。其代码如下：

```
interface Two extends One{                      //声明接口,并实现接口之间的继承
    function FunTwo();                          //声明抽象方法
}
```

2. 接口的应用

因为接口不能进行实例化的操作，所以要使用接口中的成员，那么就必须借助子类。在子类中继承接口使用 implements 关键字。如果要实现多个接口的继承，那么每个接口之间需使用逗号“,” 连接。

提示：

既然通过子类继承了接口中的方法，那么接口中的所有方法必须都在子类中实现，否则 PHP 将显示如图 13.24 所示的错误信息。

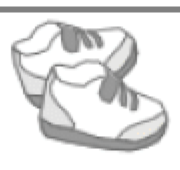
```
Fatal error: Class Member contains 2 abstract methods and
must therefore be declared abstract or implement the
remaining methods (Popedom::setPopedom,
Popedom::getPopedom) in F:\PkhPHP\www\MR\Instance\12\12.8
\index.php on line 27
```

图 13.24　接口中方法没有在子类中全部实现

下面看一个接口的实际应用。

例 13.8　在本例中，首先声明两个接口 Person 和 Popedom。然后在子类 Member 中继承接口并声明在接口中定义的方法。最后实例化子类，调用子类中方法输出数据。代码如下：（**实例位置：配套资源\mr\13\example\13.8**）

```
<?php
    interface Person{                           //定义Person接口
        public function say();                  //定义接口方法
```




Note

```
        }
        interface Popedom{                                  //定义Popedom接口
            public function money();                        //定义接口方法
        }
        class Member implements Person,Popedom{             //类Member实现接口Person和Popedom接口
            public function say(){                          //定义say方法
                echo "我只是一名普通员工，";                 //输出信息
            }
            public function money(){                        //定义money方法
                echo "我一个月的薪水是10000元";             //输出信息
            }
        }
        $man=new Member ();                                 //实例化对象
        $man->say();                                        //调用say方法
        $man->money();                                      //调用money方法
    ?>
```

运行结果如图 13.25 所示。



图 13.25　应用接口

13.7　面向对象的多态性

视频讲解：配套资源\mr\13\video\面向对象的多态性.exe

面向对象编程的特点之三是多态性，是指一段程序能够处理多种类型对象的能力。例如，在介绍面向对象特点时举的火车和汽车的例子，虽然火车和汽车都可以移动，但是它们的行为是不同的，火车要在铁轨上行驶，而汽车则在公路上行驶。

在 PHP 中，多态有两种实现方法：通过继承实现多态和通过接口实现多态。

13.7.1　通过继承实现多态

继承性已经在前面讲解过，这里直接给出一个实例，展示通过继承实现多态的方法。

例 13.9　首先创建一个抽象类 type，用于表示各种交通方法，然后让子类继承这个 type 类。代码如下：**（实例位置：配套资源\mr\13\example\13.9）**

```
    <?php
        abstract class Type{                                //定义抽象类Type
            abstract function go_Type();                    //定义抽象方法go_Type()
        }
        class Type_car extends Type{                        //小轿车类继承Type抽象类
```





```
            public function go_Type(){                  //重写抽象方法
                echo "我开着小轿车去拉萨";                //输出信息
            }
        }
        class Type_bus extends Type{                    //定义巴士车继承Type类
            public function go_Type(){                  //重写抽象方法
                echo "我坐巴士去拉萨";
            }
        }
        function change($obj){          //自定义方法根据传入对象不同调用不同类中的方法
            if($obj instanceof Type){
                $obj->go_Type();
            }else{
                echo "传入的参数不是一个对象";            //输出信息
            }
        }
        echo "实例化Type_car：";
        change(new Type_car());                         //实例化Type_car类
        echo "<br>";
        echo "实例化Type_bus：";
        change(new Type_bus);                           //实例化Type_bus类
    ?>
```

运行结果如图 13.26 所示。

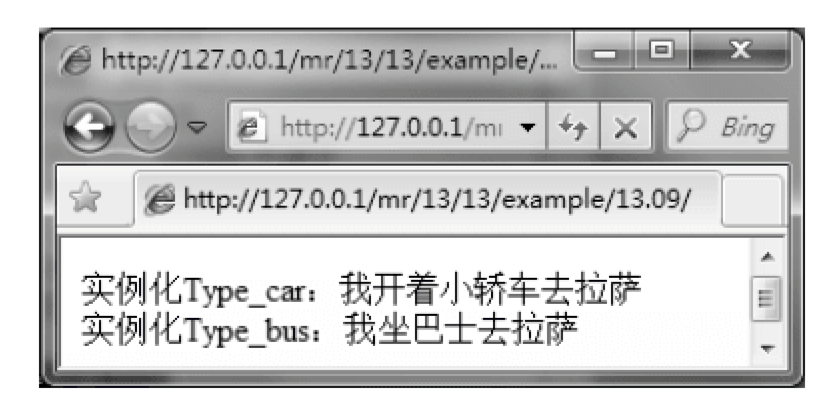


图 13.26　通过继承实现多态

在上述实例中对于抽象类 Vehicle 而言，Train 类和 Car 类就是其多态性的体现。

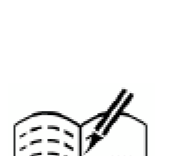
13.7.2　通过接口实现多态

下面通过实例讲解如何通过接口实现多态。

例 13.10　在本例中，首先定义接口 Type，并定义一个空方法 go_type()。然后定义 Type_car 和 Type_bus 子类继承接口 Type。最后通过 instanceof 关键字检查对象是否属于接口 Type。代码如下：（**实例位置：配套资源\mr\13\example\13.10**）

```
    <?php
        interface Type{                                 //定义Type接口
            public function go_Type();                  //定义接口方法
        }
        class Type_car implements Type{                 //Type_car类实现Type接口
            public function go_Type(){                  //定义go_Type()方法
                echo "我开着小轿车去拉萨";                //输出信息
```






```
        }
    }
    class Type_bus implements Type{                    //Type_bus实现Type方法
        public function go_Type(){                     //定义go_Type方法
            echo "我坐巴士去拉萨";                        //输出信息
        }
    }
    function change($obj){                             //自定义方法
        if($obj instanceof Type){
            $obj->go_Type();
        }else{
            echo "传入的参数不是一个对象";                  //输出信息
        }
    }
    echo "实例化Type_car：";
    change(new Type_car);                              //实例化对象
    echo "<br>";
    echo "实例化Type_bus：";
    change(new Type_bus);
?>
```

其运行结果与例 13.9 相同。

13.8　面向对象的关键字

视频讲解：配套资源\mr\13\video\面向对象的关键字.exe

13.8.1　final 关键字

final 的中文含义是最终的、最后的。被 final 修饰过的类和方法就是“最终的版本”。如果有一个类的格式为：

```
final class class_name{
//…
}
```

说明该类不可以再被继承，也不能再有子类。

如果有一个方法的格式为：

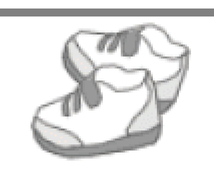
```
final function method_name()
```

说明该方法在子类中不可以进行重写，也不可以被覆盖。

这就是 final 关键字的作用。

13.8.2　static 关键字——声明静态类成员

在 PHP 中，通过 static 关键字修饰的成员属性和成员方法被称为静态属性和静态方法。静态属性和静态方法不需要在被类实例化的情况下就可以直接使用。





1. 静态属性

静态属性就是使用 static 关键字修饰的成员属性，它属于类本身而不属于类的任何实例。它相当于存储在类中的全局变量，可以在任何位置通过类来访问。静态属性访问的语法如下：

```
类名称::$静态属性名称
```

其中的符号“::”被称为范围解析操作符，用于访问静态成员、静态方法和常量，同时还可以用于覆盖类中的成员和方法。

如果要在类内部的成员方法中访问静态属性，那么只需在静态属性的名称前加上操作符“self::”即可。

2. 静态方法

静态方法就是通过关键字 static 修改的成员方法。由于它不受任何对象的限制，所以可以不通过类的实例化直接引用类中的静态方法。静态方法引用的语法如下：

```
类名称::静态方法名称([参数1,参数2,……])
```

同样，如果要在类内部的成员方法中引用静态方法，那么也是直接在静态方法的名称前加上操作符“self::”即可。

注意：

在静态方法中只能调用静态变量，而不能调用普通变量，但普通方法却可以调用静态变量。

使用静态成员除了可以不需要实例化对象，另一个作用就是在对象被销毁后，仍然保存被修改的静态数据，以便下次继续使用。

例 13.11 首先，声明一个静态变量$num，声明一个方法，在方法的内部调用静态变量并为变量值加 1。然后，实例化类中的对象。最后，调用类中的方法。代码如下：（**实例位置：配套资源\mr\13\example\13.11**）

```
<?php
    class Web{
        static $num="0";                          //定义静态变量
        public function change(){                 //定义change方法
            echo "您是本站第".self::$num."位访客.\t";//输出静态变量信息
            self::$num++;                         //静态变量做自增运算
        }
    }
    $web=new Web();                               //实例化对象
    echo "第一次实例化调用：<br>";
    $web->change();                               //对象调用
    $web->change();
    $web->change();
    echo "<br>第二次实例化调用<br>";
    $web_wap=new Web();                           //改变对象句柄实例化对象
    $web_wap->change();
    $web_wap->change();
?>
```

运行结果如图 13.27 所示。



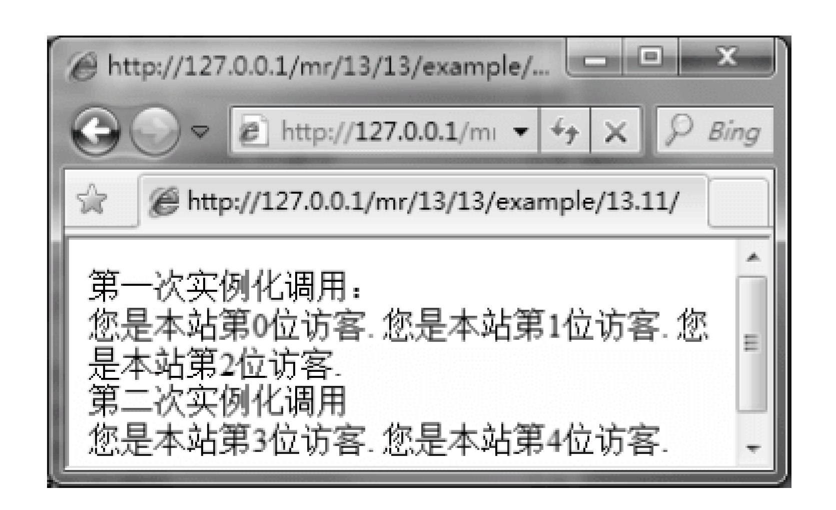


图 13.27　静态变量的使用

此时如果将程序代码中的静态变量改为普通变量，如 private $num = 0;，那么结果就不一样了。读者可以动手试一试。

指点迷津：

静态成员不用实例化对象，当类第一次被加载时就已经分配了内存空间，所以直接调用静态成员的速度要快一些。但如果静态成员声明得过多，空间一直被占用，反而会影响系统的功能。这个尺度只能通过实践积累才能真正地把握。

13.8.3　clone 关键字——克隆对象

1. 克隆对象

对象的克隆可以通过关键字 clone 来实现。使用 clone 克隆的对象与原对象没有任何关系，它是将原对象从当前位置重新复制了一份，也就是相当于在内存中重新开辟了一块空间。clone 关键字克隆对象的语法格式如下：

```
$克隆对象名称=clone $原对象名称;
```

对象克隆成功后，它们中的成员方法、属性以及值是完全相同的。如果要为克隆后的副本对象在克隆时重新为成员属性赋初始值，那么就要使用下面将要介绍的魔术方法__clone。

2. 克隆副本对象的初始化

魔术方法__clone 可以为克隆后的副本对象重新初始化。它不需要任何参数，其中自动包含$this 和$that 两个对象的引用，$this 是副本对象的引用，$that 则是原本对象的引用。

例 13.12　在对象$book1 中创建__clone 方法，将变量$object_type 的默认值从 book 修改为 computer。使用对象$book1 克隆出对象$book2，输出$book1 和$book2 的$object_type 值。代码如下：**（实例位置：配套资源\mr\13\example\13.12）**

```
<?php
class Book{                                         //类Book
    private $object_type = 'book';                  //声明私有变量$object_type，并赋初值为book
    public function setType($type){                 //声明成员方法setType，为变量$object_type赋值
        $this -> object_type = $type;
    }
    public function getType(){                      //声明成员方法getType，返回变量$object_type的值
        return $this -> object_type;
    }
    public function __clone(){                      //声明__clone方法
        $this ->object_type = 'computer';           //将变量$object_type的值修改为computer
```





```
        }
    }
    $book1 = new Book();                        //实例化对象$book1
    $book2 = clone $book1;                      //使用普通数据类型的方法给对象$book2赋值
    echo '对象$book1的变量值为：'.$book1 -> getType();          //输出对象$book1的值
    echo '<br>';
    echo '对象$book2的变量值为：'.$book2 -> getType();
    ?>
```

运行结果如图 13.28 所示。

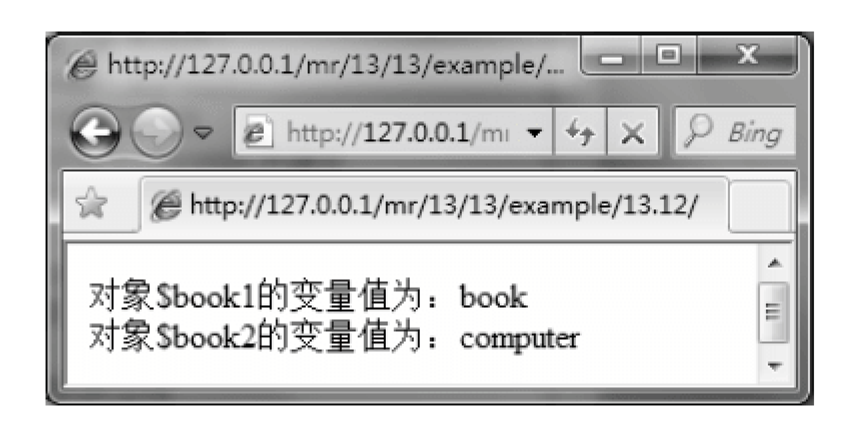


图 13.28　__clone 方法

对象$book2 克隆了对象$book1 的全部行为及属性，而且还拥有属于自己的成员变量值。

上机演练

上机演练 11　使用 final 关键字防止类被继承

使用 final 关键字防止类被继承。首先定义水果类 Fruit，然后定义 final 型的苹果类 Apple，使之继承自水果类，通过 Apple 类即可实现对苹果属性的设置。运行结果如图 13.29 所示，首先在文本框中输入苹果的颜色和形状属性，然后单击“提交”按钮即可在页面中打印出苹果的属性信息。如果再定义一个类，并使之继承 final 型的 Apple 类，则会在页面中打印出如图 13.30 所示的错误提示。

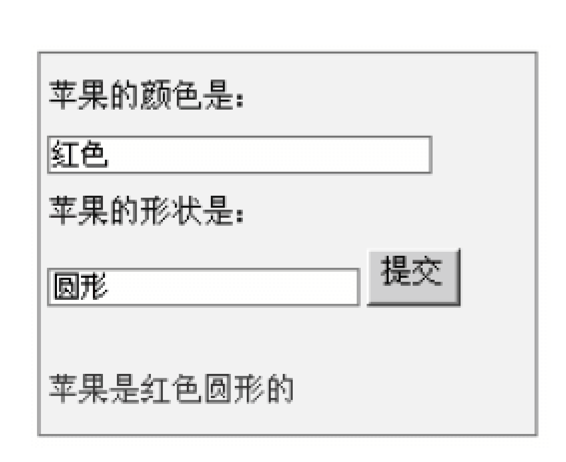


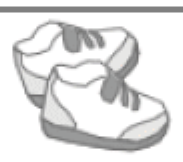
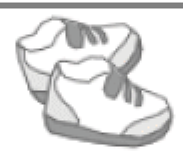
图 13.29　打印苹果的属性



图 13.30　继承 final 型水果类的错误提示

上机演练 12　使用 static 关键字定义类的静态成员

通过 static 关键字定义类的静态成员，运行结果如图 13.31 所示。首先在图中的文本框中输入要计算的数字，同时在计算类型列表框中选择计算类型，然后单击“求值”按钮，即可在页面中打印出计算结果，此时需注意的是，在进行除运算时，如果除数为 0，则应该给出错误提示，如图 13.32 所示。

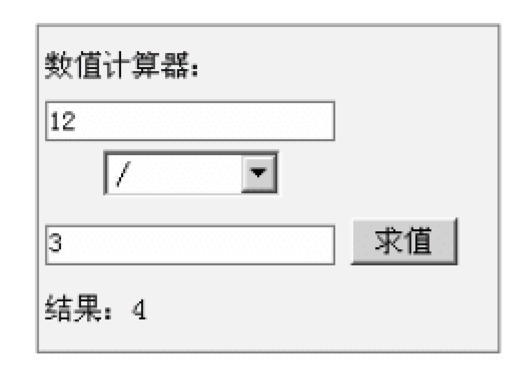




图 13.31　数值计算器

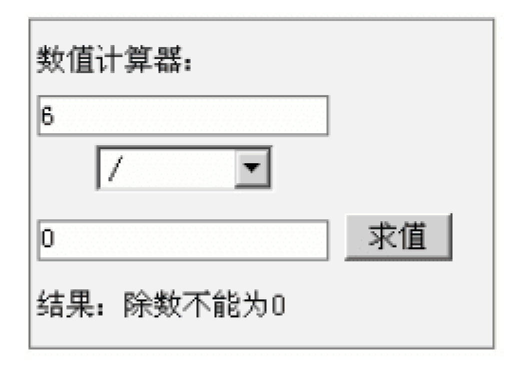


图 13.32　除数为 0 时的提示信息

上机演练 13　使用 clone 关键字实现对象的克隆

通过 clone 关键字实现对象的克隆，运行结果分别如图 13.33 和图 13.34 所示，其中图 13.33 中的输出结果是使用 clone 关键字克隆对象后，分别用原来对象和克隆的对象对类中方法进行重新赋值并调用的结果，而图 13.34 为使用等号赋值的方式来产生一个新对象后，然后再分别使用新旧对象实现对类中的方法进行调用的结果。

羊的颜色是白色

羊的颜色是灰色

图 13.33　使用 clone 关键字传递对象

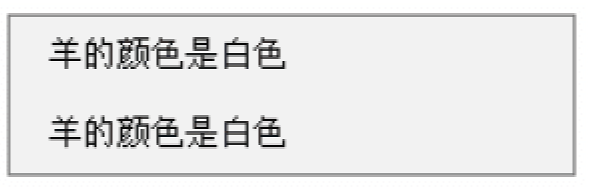


图 13.34　使用等号传递对象

13.9　面向对象的魔术方法

视频讲解：配套资源\mr\13\video\面向对象的魔术方法.exe

PHP 中有很多以两个下划线开头的方法，例如前面已经介绍过的__construct、__destruct 和__clone 方法，这些方法被称为魔术方法。

13.9.1　__set 和__get 方法

__set 和__get 方法对私有成员进行赋值或获取值的操作。

☑　__set 方法：在程序运行过程中为私有的成员属性设置值，它不需要任何返回值。__set 方法包含两个参数，分别表示变量名称和变量值。两个参数不可省略。该方法不需要主动调用，可以在方法前加上 private 关键字修饰，防止用户直接调用。

☑　__get 方法：在程序运行过程中，在对象的外部获取私有成员属性的值。它有一个必要参数，即私有成员属性名，它返回一个允许对象在外部使用的值。该方法同样不需要主动调用，可以在方法前加上 private 关键字，防止用户直接调用。

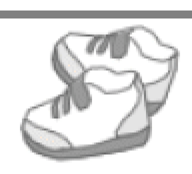
13.9.2　__isset 和__unset 方法

如果忽略__isset 和__unset 方法前面的“__”符号，用户一定会想到 isset()和 unset()函数。

isset()函数用于检测变量是否存在，如果存在则返回 True，否则返回 False。而在面向对象中，通过 isset()函数可以对公有的成员属性进行检测，但是对于私有的成员属性，这个函数就不起作用了，而魔术方法__isset 的作用就是帮助 isset()函数检测私有成员属性。

如果在对象中存在__isset 方法，当在类的外部使用 isset()函数检测对象中的私有成员属性时，就会自动调用类中的__isset 方法完成对私有成员属性的检测操作。其语法如下：

```
bool_isset(string name)          //传入对象中的成员属性名，返回值为测定结果
```





unset()函数的作用是删除指定的变量，参数为要删除的变量名称。而在面向对象中，通过unset()函数可以对公有的成员属性进行删除操作，但是对于私有的成员属性，那么就必须有__unset 方法的帮助才能够完成。

__unset 方法帮助 unset()函数在类的外部删除指定的私有成员属性。其语法格式如下：

```
void_unset(string name)        //传入对象中的成员属性名，执行将私有成员属性删除的操作
```

13.9.3　__call 方法

__call 方法的作用是：当程序试图调用不存在或不可见的成员方法时，PHP 会先调用__call 方法来存储方法名及其参数。__call 方法包含两个参数，即方法名和方法参数。其中，方法参数是以数组形式存在的。

例 13.13　本例中声明一个类 MrSoft，包含两个方法：MingRi 和__call。类实例化后，调用一个不存在的方法 MingR，看魔术方法__call 的妙用。代码如下：（**实例位置：配套资源\mr\13\example\13.13**）

```
<?php
class MrSoft{
    public function MingRi(){                          // MingRi方法
        echo '调用的方法存在，直接执行此方法。<p>';
    }
    public function __call($method, $parameter) {      //__call方法
        echo '如果方法不存在，则执行__call()方法。<br>';
        echo '方法名为：'.$method.'<br>';              //输出第一个参数，即方法名
        echo '参数有：';
        var_dump($parameter);                          //输出第二个参数，是一个参数数组
    }
}
$mrsoft = new MrSoft();                                //实例化对象$mrsoft
$mrsoft -> MingRi();                                   //调用存在的MingRi方法
$mrsoft -> MingR('how','what','why');                  //调用不存在的MingR方法
?>
```

运行结果如图 13.35 所示。

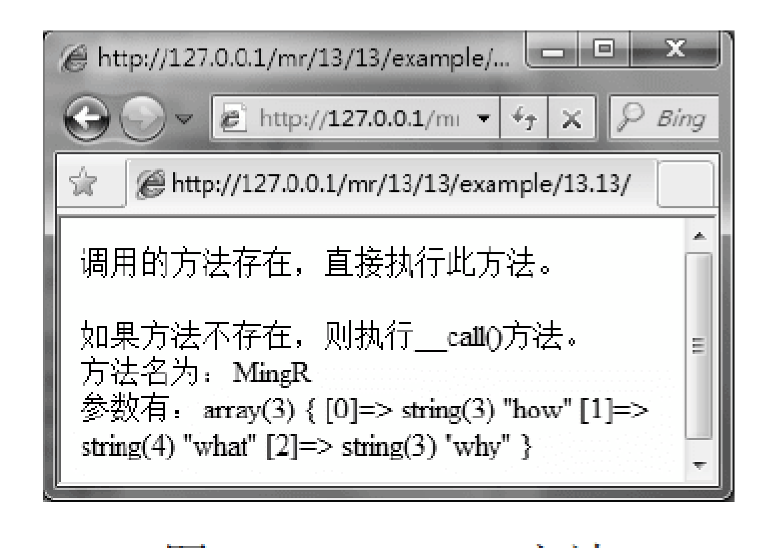


图 13.35　__call 方法

13.9.4　__toString 方法

魔术方法__toString 的作用是：当使用 echo 或 print 输出对象时，将对象转化为字符串。

例 13.14　定义 People 类，应用__toString 方法输出 People 类的实例化对象$peo。代码如





Note

下：（**实例位置：配套资源\mr\13\example\13.14**）

```
<?php
    class People{
        public function __toString(){
            return "我是toString的方法体";
        }
    }
    $peo=new People();
    echo $peo;
?>
```

运行结果为：我是 toString 的方法体

指点迷津：

（1）如果没有__toString 方法，直接输出对象将会发生致命错误（fatal error）。

（2）在输出对象时必须注意的是，echo 或 print 函数后面应直接跟要输出的对象，中间不要加多余的字符，否则__toString 方法不会被执行，例如 echo '字串'.$myComputer、echo ''.$myComputer 等。

13.9.5 __autoload 方法

将一个独立、完整的类保存到一个 PHP 页中，并且文件名和类名保持一致，这是每个开发人员都需要养成的良好习惯。这样，在下次重复使用某个类时就可以很轻松地找到它。但还有一个问题让开发人员头疼不已，即：如果要在一个页面中引进很多的类，需要使用 include_once()函数或 require_once()函数一个一个地引入。

在 PHP5 中应用__autoload 方法解决了这个问题。__autoload 方法可以自动实例化需要使用的类。当程序要用到一个类，但该类还没有被实例化时，PHP5 将使用__autoload 方法，在指定的路径下自动查找和该类名称相同的文件。如果找到则继续执行，否则报告错误。

例 13.15 首先创建一个类文件 inc.php，该文件包含类 People。然后创建 index.php 文件，在文件中创建__autoload 方法，判断类文件是否存在，如果存在则使用 require_once 函数将文件动态引入，否则输出提示信息。类文件 inc.php 的代码如下：（**实例位置：配套资源\mr\13\example\13.15**）

```
<?php
    class People{                              //定义类
        public function __toString(){          //定义__toString方法
            return"自动加载类";
        }
    }
?>
```

index.php 文件的代码如下：

```
<?php
    function __autoload($class_name){          //创建__autoload方法
        $class_path = $class_name.'/inc.php';  //类文件路径
        if(file_exists($class_path)){          //判断类文件是否存在
```



```
                include_once($class_path);
                    //动态包含类文件
            }else
                echo '类路径错误。';
        }
        $mrsoft = new People();                        //实例化对象
        echo $mrsoft;                                  //输出类内容
    ?>
```

运行结果为：自动加载类。

上机演练

上机演练 14　使用__set 方法为类中未经定义的属性赋值

通过__set 方法存取类中未声明的属性。运行效果如图 13.36 所示，在页面中以表格的形式列出各本图书的详细信息，其中图书的“备注”属性是使用__set 方法所赋的值。

书　名	页　码	作　者	价　格	备　注
《PHP从基础到**》	650	小张、小潘、小王	58	备注
《PHP函数**》	800	小潘、小王	80	备注
《PHP范例**》	700	小李、小懂	85	备注
《PHP实战**》	750	小郭、小刘	75	备注

图 13.36　图书信息列表

上机演练 15　使用__get 方法获取未声明属性的名称

通过__get 方法获得类中未定义属性的值。运行效果如图 13.37 所示，页面中打印的内容是通过调用苹果类的属性打印的结果，而弹出的提示对话框是因为没有在类中定义$produceArea 属性而通过__get 方法弹出的。



图 13.37　类中未定义属性提示框

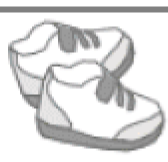
本 章 摘 要

本章主要讲解面向对象编程技术，围绕面向对象的封装性、继承性和多态性 3 个特点进行详细讲解。

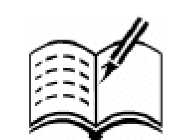
1．类的声明。

2．类的实例化。

3．类的封装、继承和多态性。



4．抽象类和接口的应用。
5．面向对象的关键字。
6．面向对象的魔术方法。



Note

习　　题

1．实现类之间的继承需要使用哪个关键字？（　　）
A．public　　B．set　　C．extends
2．构造方法需要使用哪个关键字？（　　）
A．__construct　　B．__destruct　　C．__set　　D．__get
3．使类不能被其他子类所继承需要使用哪个关键字？（　　）
A．private　　B．static　　C．final
4．声明接口需要使用哪个关键字？（　　）
A．class　　B．abstract　　C．Interface
5．PHP 中的类继承可以（　　）。
A．多继承　　B．单继承　　C．既可以单继承也可以多继承
6．将类 People 声明为抽象类的代码为（　　）。
7．克隆对象需要使用（　　）关键字。
8．声明静态变量需要使用（　　）关键字。
9．检测某个对象是否从属于某个类需要使用（　　）关键字。
10．程序运行后，页面中显示的内容为（　　）。

```
class Class_name {
        function __construct() {
            echo "我是初始化函数，在实例化的同时就被调用了";
        }
    }
new Class_name();
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 12 章内容，其中对所列实例的实现方法进行了详细讲解）。

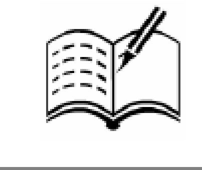
实战模拟 1　使用单例模式制作数据库管理类

使用单例模式制作一个基于 MySQL 数据库的数据库管理类。运行效果如图 13.38 所示，首先在用户登录文本框中输入用户名和密码，然后单击表单中的“登录”按钮。如果用户名和密码经后台代码验证正确，则当前页面将跳转到登录成功信息提示页面；如果错误则在页面打印错误提示。注意，在上述过程中，对数据库管理模块在技术上采用 PDO 技术，而在设计上是采用单例模式进行编码的。





图 13.38　用户登录表单



实战模拟 2　使用策略模式打印客户端浏览器类型

通过策略模式输出浏览器的类型。运行效果分别如图 13.39 和图 13.40 所示，其中图 13.39 中的结果是在 IE 下运行的结果。图 13.40 所示为在火狐浏览器下运行的结果，从运行结果可知，可以根据不同浏览器的类型在页面中打印相应浏览器的名称。



图 13.39　打印浏览器的类型为 IE



图 13.40　打印浏览器的类型是 Firefox

实战模拟 3　使用工厂模式设置用户访问权限

本例通过工厂模式对用户权限进行管理。运行效果如图 13.41 所示，在页面中将打印出不同用户的类型及各项操作的访问权限。制作本范例时，首先创建抽象的用户权限类，然后分别定义浏览、添加、更改和删除等子类，使之继承基类用户权限类，然后根据不同用户的权限，重写用户权限类中的方法，最后创建用户权限工厂来根据不同用户类别返回相应的对象。



图 13.41　电子相册用户权限列表

▶▶第 3 篇

高级应用篇



PDO 数据库抽象层

（ 自学视频、源程序：配套资源\mr\14\ ）

在 PHP 的早期版本中，各种不同的数据库扩展（MySQL、MS SQL、Oracle）根本没有真正的一致性，虽然它们都可以实现相同的功能，但是这些扩展却互不兼容，都有各自的操作函数，各自为政。结果导致 PHP 的维护非常困难，可移植性也非常差，为了解决这些问题，PHP 开发人员编写了一种轻型、便利的 API 来统一各种数据库的共性，从而达到 PHP 脚本最大程度的抽象性和兼容性，这就是数据库抽象层。而在本章中将要介绍的是目前 PHP 抽象层中最为流行的一种——PDO 数据库抽象层。

学习摘要：

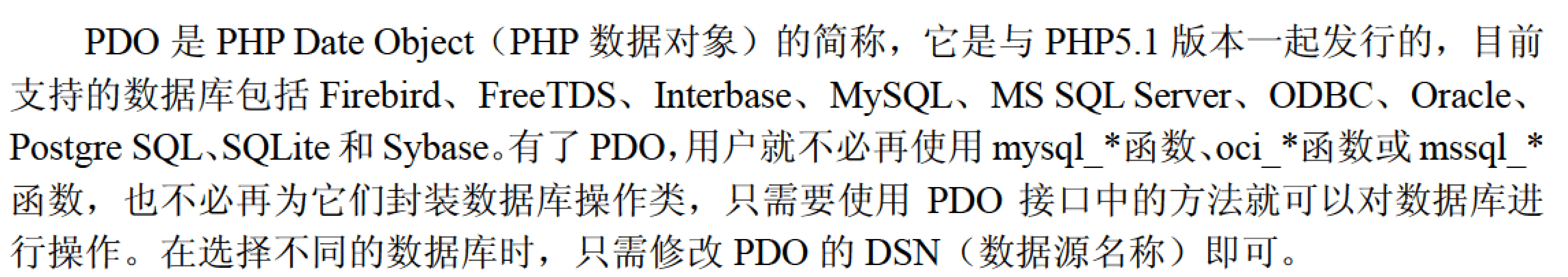
- 了解 PDO
- PHP 的安装、配置
- PDO 连接 MySQL、MS SQL Server、Oracle 数据库
- PDO 中 exec、query 和预处理语句执行 SQL 语句
- PDO 中 fetch、fetchAll 或 fetchColumn 方法获取结果集
- PDO 中捕获 SQL 语句中的错误
- PDO 中的错误处理
- PDO 中的事务处理
- PDO 中的存储过程





14.1　什么是 PDO

14.1.1　PDO 概述



PDO 是 PHP Date Object（PHP 数据对象）的简称，它是与 PHP5.1 版本一起发行的，目前支持的数据库包括 Firebird、FreeTDS、Interbase、MySQL、MS SQL Server、ODBC、Oracle、Postgre SQL、SQLite 和 Sybase。有了 PDO，用户就不必再使用 mysql_*函数、oci_*函数或 mssql_*函数，也不必再为它们封装数据库操作类，只需要使用 PDO 接口中的方法就可以对数据库进行操作。在选择不同的数据库时，只需修改 PDO 的 DSN（数据源名称）即可。

在 PHP6 中将默认使用 PDO 连接数据库，所有非 PDO 扩展将会在 PHP6 中被移除。该扩展提供 PHP 内置类 PDO 来对数据库进行访问，不同的数据库使用相同的方法名，从而解决数据库连接不统一的问题。

14.1.2　PDO 特点

PDO 是一个“数据库访问抽象层”，其作用是统一各种数据库的访问接口，与 MySQL 和 MsSQL 函数库相比，PDO 让跨数据库的使用更具有亲和力；与 ADODB 和 MDB2 相比，PDO 更高效。

PDO 将通过一种轻型、清晰、方便的函数，统一各种不同 RDBMS 库的共有特性，实现 PHP 脚本最大程度的抽象性和兼容性。

PDO 吸取现有数据库扩展成功和失败的经验教训，利用 PHP5 的最新特性，可以轻松地与各种数据库进行交互。

PDO 扩展是模块化的，使用户能够在运行时为数据库后端加载驱动程序，而不必重新编译或重新安装整个 PHP 程序。例如，PDO_MySQL 扩展会替代 PDO 扩展实现 MySQL 数据库 API。另外还有一些用于 Oracle、PostgreSQL、ODBC 和 Firebird 的驱动程序，更多的驱动程序尚在开发。

14.1.3　安装 PDO

PDO 是与 PHP5.1 一起发行的，默认包含在 PHP5.1 中。由于 PDO 需要 PHP5 核心面向对象特性的支持，因此其无法在 PHP5.0 之前的版本中使用。

默认情况下，PDO 在 PHP5.2 中为开启状态，但是要启用对某个数据库驱动程序的支持，仍需要进行相应的配置操作。

在 Linux 环境下，要使用 MySQL 数据库，可以在 configure 命令中添加如下选项：

```
--with-pdo-mysql=/path/to/mysql/installation
```

在 Windows 环境下，PDO 在 php.ini 文件中进行配置，如图 14.1 所示。

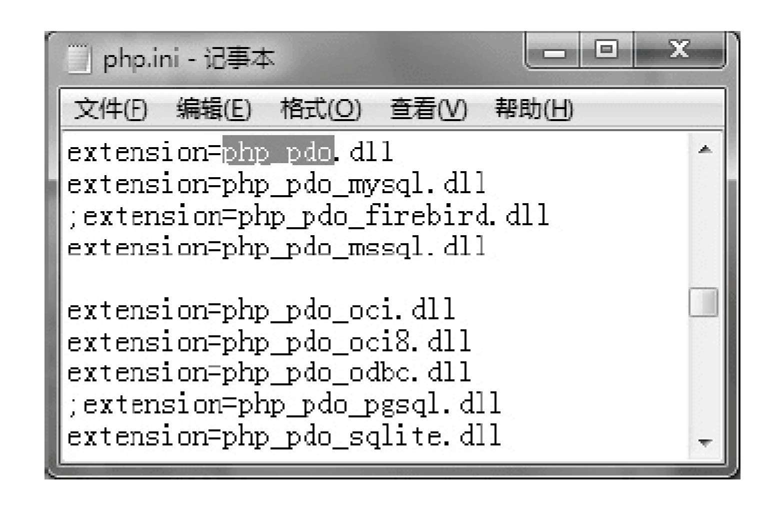
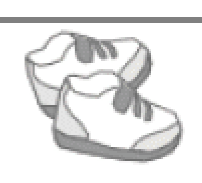


图 14.1　在 Windows 环境下配置 PDO



要启用 PDO，首先必须加载 extension=php_pdo.dll 选项，如果要想其支持某个具体的数据库，那么还要加载对应的数据库选项。例如，要支持 MySQL 数据库，则需要加载 extension=php_pdo_mysql.dll 选项。



Note

脚下留神：

在完成数据库的加载后，要保存 php.ini 文件，并且重新启动 Apache 服务器，修改才能生效。

14.2　PDO 连接数据库

视频讲解：配套资源\mr\14\video\PDO 连接数据库.exe

14.2.1　PDO 构造函数

在 PDO 中，要建立与数据库的连接需要实例化 PDO 的构造函数。PDO 构造函数的语法如下：

```
__construct(string $dsn[,string $username[,string $password[,array $driver_options]]])
```

参数 dsn：数据源名，包括主机名端口号和数据库名称；参数 username：连接数据库的用户名；参数 password：连接数据库的密码；参数 driver_options：连接数据库的其他选项。

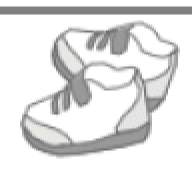
通过 PDO 连接 MySQL 数据库的代码如下：

```
<?php
    $dbms='mysql';                                          //数据库类型
    $dbName='db_database14';                                //使用的数据库名称
    $user='root';                                           //使用的数据库用户名
    $pwd='111';                                             //使用的数据库密码
    $host='localhost';                                      //使用的主机名称
    $dsn="$dbms:host=$host;dbname=$dbName";
    try {                                                   //捕获异常
        $pdo=new PDO($dsn,$user,$pwd);                      //实例化对象
        echo "PDO连接MySQL成功";
    } catch (Exception $e) {
        echo $e->getMessage()."<br>";
    }
?>
```

14.2.2　DSN 详解

DSN 是 Data Source Name（数据源名称）的首字母缩写，它提供连接数据库需要的信息。PDO 的 DSN 包括 3 部分：PDO 驱动名称（例如 mysql、sqlite 或 pgsql）、冒号和驱动特定的语法。每种数据库都有其特定的驱动语法。

在使用不同的数据库时，必须明确数据库服务器是完全独立于 PHP 的是实体。虽然笔者在讲解本书的内容时，数据库服务器和 Web 服务器是在同一台计算机上，但是实际的情况可能不是如此。数据库服务器可能与 Web 服务器不是在同一台计算机上，此时要通过 PDO 连接数据库时，就需要修改 DSN 中的主机名称。



由于数据库服务器只在特定的端口上监听连接请求。每种数据库服务器具有一个默认的端口号（MySQL 是 3306），但是数据库管理员可以对端口号进行修改，因此有可能 PHP 找不到数据库的端口，此时就可以在 DSN 中包含端口号。

另外，由于一个数据库服务器中可能拥有多个数据库，所以在通过 DSN 连接数据库时，通常都包括数据库名称，这样可以确保连接的是用户想要的数据库，而不是其他人的数据库。



Note

上机演练

上机演练 1　连接 MySQL 数据库

在学习通过 PDO 连接 MySQL 数据库之前，大家先在 phpMyAdmin 下创建一个 MySQL 数据库 db_database14，并且在 db_database14 数据库中创建数据表 tb_pdo，然后定义数据库连接的参数，最后，通过 PDO 构造函数创建连接。其运行效果如图 14.2 所示。

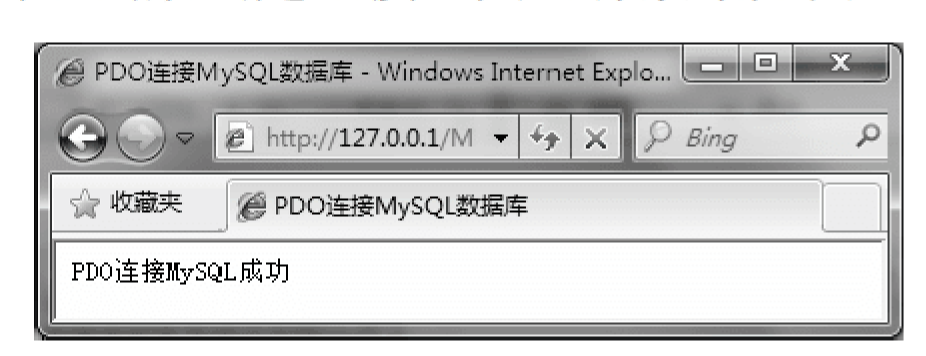


图 14.2　连接 MySQL 数据库

上机演练 2　连接 MS SQL Server 数据库

本例连接 MS SQL Server 数据库服务器，数据库名称是 db_database14，数据表是 tb_pdo_mssql。数据库服务器是 PC-201006101638，数据库服务器的用户名是 sa，密码为空。

连接 MS SQL Server 与连接 MySQL 数据库的不同之处就是指定不同的数据库类型，即 MS SQL Server 的数据库类型是 mssql。

指点迷津：

（1）如何测试你的 PHP 是否支持 MS SQL Server 数据库。编写一个 phpinfo.php 文件，文件内容如下：<?php echo phpinfo(); ?>,运行该文件，即可查看出 PHP 是否支持 MS SQL Server 数据库。

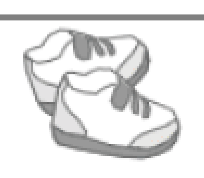
（2）在 php.ini 文件中有关 mssql 的配置已经修改完成后，PHP 仍不支持 mssql 的解决方案：将 PHP 安装目录下的 ntwdblib.dll 文件复制到本机系统盘的 WINDOWS\system32 文件夹下，然后重新启动 Apache 服务器

上机演练 3　连接 Oracle 数据库

本例连接 Oracle 数据库服务器，数据库名称是 192.168.1.59:1521/oralcles，数据库服务器的用户名是 system，密码是 mrsoft。

指点迷津：

通过 PDO 连接 Oracle 数据库，首先要在本机上安装 Oracle 数据库的客户端。然后，去掉 php.ini 文件中 extension=php_pdo.dll 和 extension=php_pdo_oci.dll 前面的分号，加载 PDO 模块。最后，将 Oracle 客户端的 oraociei10.dll、oci.dll 和 orannzsbb10.dll 复制到 Apache 的 bin 文件夹下。





Note

14.3　PDO 中执行 SQL 语句

视频讲解：配套资源\mr\14\video\PDO 中执行 SQL 语句.exe

在 PDO 中，可以使用下面的 3 种方法来执行 SQL 语句。

14.3.1　exec 方法

exec 方法返回执行后受影响的行数，其语法如下：

```
int PDO::exec ( string statement )
```

参数 statement 是要执行的 SQL 语句。该方法返回执行查询时受影响的行数，通常用于 INSERT、DELETE 和 UPDATE 语句中。

例 14.1　使用 exec 方法执行删除操作，具体步骤如下。（**实例位置：配套资源\mr\14\example\14.1**）

创建 index.php 文件，设计网页页面。首先，通过 PDO 连接 MySQL 数据库。然后，定义 DELETE 删除语句，应用 exec 方法执行删除操作。其关键代码如下：

```
<?php
    $dbms='mysql';
    $dbName='db_database14';
    $user='root';
    $pwd='111';
    $host='localhost';
    $dsn="$dbms:host=$host;dbname=$dbName";
    $query="delete from tb_zc where id=3";//SQL语句
            try {
                $pdo=new PDO($dsn,$user,$pwd);
                $affCount=$pdo->exec($query);
                echo "删除成功，受影响条数为".$affCount;
            } catch (Exception $e) {
                echo "ERROR!!".$e->getMessage()."<br>";
            }
?>
```

运行结果如图 14.3 所示。

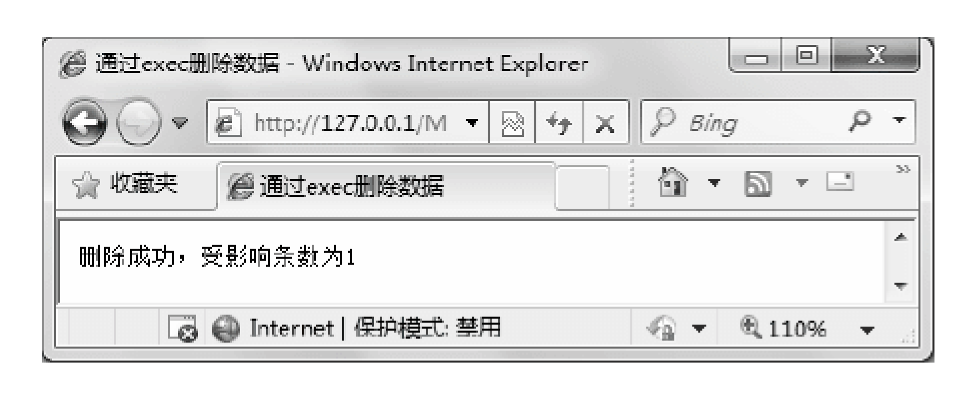


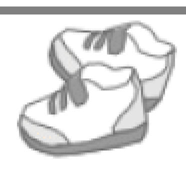
图 14.3　应用 exec()方法删除数据

14.3.2　query 方法

query 方法通过用于返回执行查询后的结果集。其语法如下：

```
PDOStatement PDO::query ( string statement )
```

参数 statement 是要执行的 SQL 语句，它返回的是一个 PDOStatement 对象。



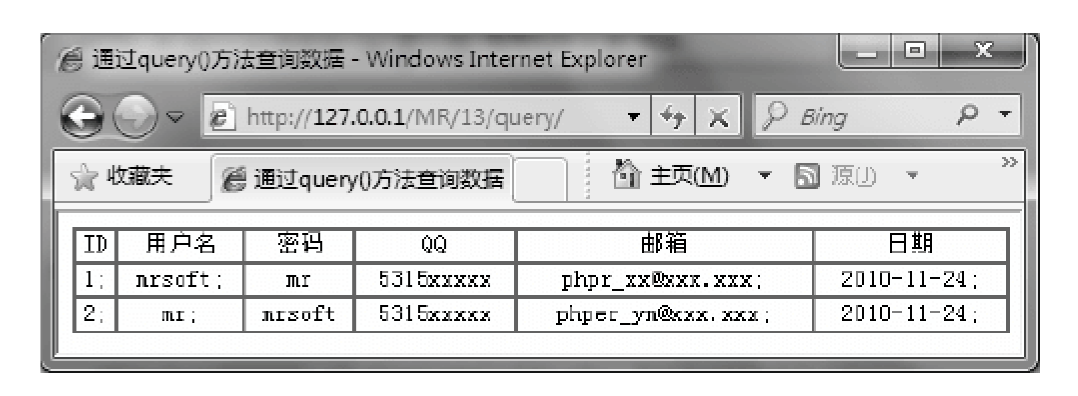


例 14.2　使用 query 方法执行查询操作，具体步骤如下。（**实例位置：配套资源\mr\14\example\14.2**）

创建 index.php 文件，设计网页页面。首先，通过 PDO 连接 MySQL 数据库。然后，通过 query 方法执行查询。最后应用 foreach()函数以表格形式输出查询内容。其关键代码如下：

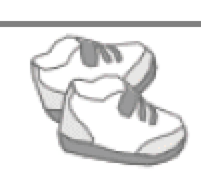
```
<table width="515" border="0" bgcolor="#FF3366">
  <tr>
    <td bgcolor="#FFFFFF"><div align="center">ID</div></td>
    <td bgcolor="#FFFFFF"><div align="center">用户名</div></td>
    <td bgcolor="#FFFFFF"><div align="center">密码</div></td>
    <td bgcolor="#FFFFFF"><div align="center">QQ</div></td>
    <td bgcolor="#FFFFFF"><div align="center">邮箱</div></td>
    <td bgcolor="#FFFFFF"><div align="center">日期</div></td>
  </tr>
<?php
    $dbms='mysql';
    $dbName='db_database14';
    $user='root';
    $pwd='111';
    $host='localhost';
    $dsn="$dbms:host=$host;dbname=$dbName";
    $query="select * from tb_zc ";//SQL语句
         try {
             $pdo=new PDO($dsn,$user,$pwd);
             $result=$pdo->query($query);          //输出结果集中的数据
             foreach ( $result as $row){           //输出结果集中的数据
?>
  <tr>
    <td bgcolor="#FFFFFF"><div align="center"><?php echo   $row['id'];?>;</div></td>
    <td bgcolor="#FFFFFF"><div align="center"><?php echo   $row['username'];?>;</div></td>
    <td bgcolor="#FFFFFF"><div align="center"><?php echo   $row['userpwd'];?></div></td>
    <td bgcolor="#FFFFFF"><div align="center"><?php echo   $row['qq'];?></div></td>
    <td bgcolor="#FFFFFF"><div align="center"><?php echo   $row['email'];?>;</div></td>
    <td bgcolor="#FFFFFF"><div align="center"><?php echo   $row['date'];?>;</div></td>
  </tr>
  <?php }
             } catch (Exception $e) {
                 echo "ERROR!!".$e->getMessage()."<br>";
                 }
  ?>
</table>
```

运行结果如图 14.4 所示。

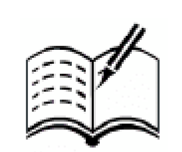


ID	用户名	密码	QQ	邮箱	日期
1;	mrsoft;	mr	5315xxxxx	phpr_xx@xxx.xxx;	2010-11-24;
2;	mr;	mrsoft	5315xxxxx	phper_yn@xxx.xxx;	2010-11-24;

图 14.4　使用 query 方法的输出结果



14.3.3　预处理语句——prepare 和 execute



预处理语句包括 prepare 和 execute 两个方法。首先，通过 prepare 方法做查询的准备工作，然后，通过 execute 方法执行查询，并且还可以通过 bindParam 方法来绑定参数提供给 execute 方法。其语法如下：

```
PDOStatement PDO::prepare ( string statement [, array driver_options] )
bool PDOStatement::execute ( [array input_parameters] )
```

例 14.3　在 PDO 中通过预处理语句 prepare 和 execute 执行 SQL 查询语句，并且应用 while 语句和 fetch 方法完成数据的循环输出，具体实现如下。（**实例位置：配套资源\mr\14\example\14.3**）

首先完成页面效果的设计。然后通过 PDO 连接 MySQL 数据库。接着通过预处理语句 prepare 和 execute 执行 SQL 查询语句。最后通过 while 语句和 fetch 方法完成数据的循环输出。其关键代码如下：

```
<?php
$dbms='mysql';              //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，
而无须记住那么多的函数
$host='localhost';                      //数据库主机名
$dbName='db_database14';                //使用的数据库
$user='root';                           //数据库连接用户名
$pass='111';                            //对应的密码
$dsn="$dbms:host=$host;dbname=$dbName";
try {
$pdo = new PDO($dsn, $user, $pass);    //初始化一个PDO对象，即创建了数据库连接对象$pdo
    $query="select * from tb_pdo_mysql";      //定义SQL语句
    $result=$pdo->prepare($query);      //准备查询语句
    $result->execute();                 //执行查询语句，并返回结果集
    while($res=$result->fetch(PDO::FETCH_ASSOC)){//循环输出查询结果集，设置结果集为关
联索引
    ?>
          <tr>
            <td height="22" align="center" valign="middle"><?php echo $res['id'];?></td>
            <td align="center" valign="middle"><?php echo $res['pdo_type'];?></td>
            <td align="center" valign="middle"><?php echo $res['database_name'];?></td>
            <td align="center" valign="middle"><?php echo $res['dates'];?></td>
          </tr>
<?php
    }
          } catch (PDOException $e) {
    die ("Error!: " . $e->getMessage() . "<br/>");
}
?>
```

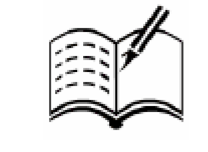
运行结果如图 14.5 所示。





ID	PDO	数据库	时间
1	pdo	mysql	2010-09-18 10:19:17
2	pdo	Oracle	2010-09-18 10:19:17
4	pdo	Access	2010-10-10 00:00:00
28	pdo	access	2011-06-25 15:00:00
27	pdo	access	2011-06-25 00:00:00
22	pdo	SQL	2010-10-09 10:59:30
23	pdo	db2	2010-10-16 10:59:41
24	pdo	mysql	2010-08-21 00:00:00
25	pdo	MySQL	2010-10-10 10:10:10
26	pdo	MySQL	2010-10-10 10:10:10

图 14.5　通过 prepare 和 execute 方法执行 SQL 查询语句



多学两招：

预处理语句是要运行的 SQL 的一种编译过的模板，它可以使用变量参数进行定制。预处理语句可以带来两大好处：

（1）查询只需解析（或准备）一次，但是可以用相同或不同的参数执行多次。当查询准备好后，数据库将分析、编译和优化执行该查询的计划。对于复杂的查询，这个过程要花比较长的时间，如果用户需要以不同的参数多次重复相同的查询，那么该过程将会大大降低应用程序的速度。通过使用预处理语句可以避免重复分析/编译/优化周期。简言之，预处理语句使用更少的资源，因而运行得更快。

（2）提供给预处理语句的参数不需要用引号括起来，驱动程序会处理这些。如果应用程序独自地使用预处理语句，那么可以确保没有 SQL 入侵发生；但如果用户仍然将查询的其他部分建立在不受信任的输入之上，那么就仍然存在风险。

指点迷津：

PDO 中执行 SQL 语句方法的选择包括：

（1）如果是只执行一次查询，那么 PDO->query 是较好的选择。虽然它无法自动转义发送给它的任何数据，但是它在遍历 SELECT 语句的结果集方面是非常方便的。然而在使用该方法时也要相当小心，因为如果没有在结果集中获取到所有数据，那么下次调用 pdo->query 时可能会失败。

（2）如果是多次执行 SQL 语句，那么最理想的方法是 prepare 和 execute。这两个方法可以对提供给它们的参数进行自动转义，进而防止 SQL 注入攻击。同时由于在多次执行 SQL 语句时，应用的是预编译语句，所以还可以减少资源的占用，提高运行速度。

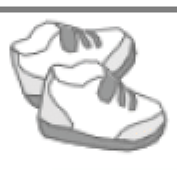
上机演练

上机演练 4　通过 PDO 向数据库中添加数据

PDO 数据库抽象层的主要特点是为不同的数据库提供统一的接口，使用户在后期维护或数据库变更时减少麻烦等。通过 PDO 向已经创建好的数据库中添加数据，其运行结果如图 14.6 所示。

上机演练 5　通过 PDO 浏览数据库中的数据

利用 PDO 抽象层显示数据需要使用 query 方法，此函数与普通的操作数据库函数并无区别，只不过在 PDO 中此函数只用于数据的查询，而插入、修改和更新需要使用 exec 方法来实现。其运行结果如图 14.7 所示。



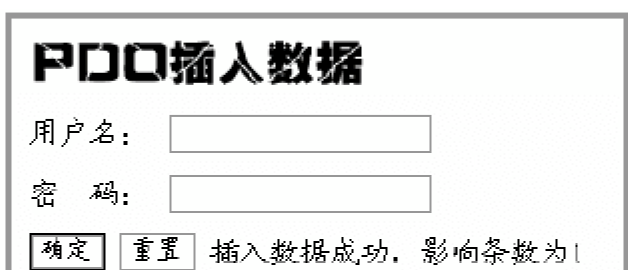


图 14.6　通过 PDO 向数据库中添加数据

ID	用户名称	用户密码	操作时间
1	mr	mrsoft	2010-11-24
2	mrsoft	mrkj	2010-11-24
3	ym	066066	2010-11-24
4	pkh	123456	2010-11-24

图 14.7　通过 PDO 浏览数据库中的数据

上机演练 6　通过 PDO 更新数据库中的数据

在利用 MySQL 数据库函数更新数据时，需要将要更改的数据信息 id 拼接到地址栏的参数中。在 PDO 中操作数据更新也不例外。虽然利用 PDO 操作数据的方法是不同的，但原理却是相同的。其运行效果如图 14.8 和图 14.9 所示。

ID	用户名称	用户密码	操作时间	修改
1	mr	mrsoft	2010-11-24	修改
2	mrsoft	mrkj	2010-11-24	修改
3	ym	066066	2010-11-24	修改
4	pkh	123456	2010-11-24	修改
3	yangming	••••••	2010-11-24	确定

图 14.8　通过 PDO 更新数据库中的数据



图 14.9　更新数据成功

14.4　PDO 中获取结果集

视频讲解：配套资源\mr\14\video\PDO 中获取结果集.exe

在 PDO 中获取结果集有 3 种方法：fetch、fetchAll 和 fetchColumn。

14.4.1　fetch 方法

fetch 方法获取结果集中的下一行，其语法格式如下：

```
mixed PDOStatement::fetch ( [int fetch_style [, int cursor_orientation [, int cursor_offset]]] )
```

参数 fetch_style：控制结果集的返回方式，其可选方式如表 14.1 所示。

表 14.1　fetch_style 控制结果集的可选值

值	说　　明
PDO::FETCH_ASSOC	关联数组形式
PDO::FETCH_NUM	数字索引数组形式
PDO::FETCH_BOTH	两者数组形式都有，这是默认的
PDO::FETCH_OBJ	按照对象的形式，类似于以前的 mysql_fetch_object
PDO::FETCH_BOUND	以布尔值的形式返回结果，同时将获取的列值赋给 bindParam 方法中指定的变量
PDO::FETCH_LAZY	以关联数组、数字索引数组和对象 3 种形式返回结果

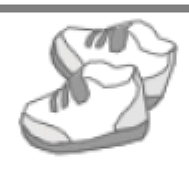
参数 cursor_orientation：PDOStatement 对象的一个滚动游标，可用于获取指定的一行。

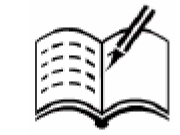
参数 cursor_offset：游标的偏移量。

例 14.4　通过 fecth 方法获取结果集中下一行的数据，进而应用 while 语句完成数据库中数据的循环输出，具体步骤如下。（**实例位置：配套资源\mr\14\example\14.4**）

创建 index.php 文件，设计网页页面。首先，通过 PDO 连接 MySQL 数据库。然后，定义 SELECT 查询语句，应用 prepare 和 execute 方法执行查询操作。接着，通过 fetch 方法返回结







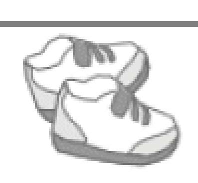
果集中下一行数据，同时设置结果集以关联数组形式返回。最后，通过 while 语句完成数据的循环输出。其关键代码如下：

```
<?php
$dbms='mysql';      //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，而无须记住那么多的函数
$host='localhost';                  //数据库主机名
$dbName='db_database14';            //使用的数据库
$user='root';                       //数据库连接用户名
$pass='111';                        //对应的密码
$dsn="$dbms:host=$host;dbname=$dbName";
try {
    $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
 $query="select * from tb_pdo_mysql";   //定义SQL语句
 $result=$pdo->prepare($query);         //准备查询语句
 $result->execute();                    //执行查询语句，并返回结果集
 while($res=$result->fetch(PDO::FETCH_ASSOC)){//循环输出查询结果集，并且设置结果集的为关联索引
 ?>
        <tr>
          <td height="22" align="center" valign="middle"><?php echo $res['id'];?></td>
          <td align="center" valign="middle"><?php echo $res['pdo_type'];?></td>
          <td align="center" valign="middle"><?php echo $res['database_name'];?></td>
          <td align="center" valign="middle"><?php echo $res['dates'];?></td>
          <td align="center" valign="middle"><a href="#">删除</a></td>
        </tr>
<?php
    }
        } catch (PDOException $e) {
    die ("Error!: " . $e->getMessage() . "<br/>");
}
      ?>
```

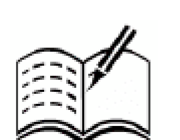
运行结果如图 14.10 所示。



图 14.10　通过 fetch 方法获取查询结果集



14.4.2 fetchAll 方法



fetchAll 方法获取结果集中的所有行。其语法如下：

```
array PDOStatement::fetchAll ( [int fetch_style [, int column_index]] )
```

参数 fetch_style：控制结果集中数据的显示方式。

参数 column_index：字段的索引。

其返回值是一个包含结果集中所有数据的二维数组。

例 14.5 通过 fecthAll 方法获取结果集中所有行，并且通过 for 语句读取二维数组中的数据，完成数据库中数据的循环输出，具体步骤如下。（**实例位置：配套资源\mr\14\example\14.5**）

创建 index.php 文件，设计网页页面。首先，通过 PDO 连接 MySQL 数据库。然后，定义 SELECT 查询语句，应用 prepare 和 execute 方法执行查询操作。接着，通过 fetchAll 方法返回结果集中所有行。最后，通过 for 语句完成结果集中所有数据的循环输出。其关键代码如下：

```
<?php
$dbms='mysql';           //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，而无须记住那么多的函数
$host='localhost';                    //数据库主机名
$dbName='db_database14';              //使用的数据库
$user='root';                         //数据库连接用户名
$pass='111';                          //对应的密码
$dsn="$dbms:host=$host;dbname=$dbName";
try {
    $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
    $query="select * from tb_pdo_mysql";              //定义SQL语句
    $result=$pdo->prepare($query);                //准备查询语句
    $result->execute();                           //执行查询语句，并返回结果集
    $res=$result->fetchAll(PDO::FETCH_ASSOC);     //获取结果集中的所有数据
    for($i=0;$i<count($res);$i++){                //循环读取二维数组中的数据
    ?>
        <tr>
                <td height="22" align="center" valign="middle"><?php echo $res[$i]['id'];?></td>
                <td align="center" valign="middle"><?php echo $res[$i]['pdo_type'];?></td>
                <td align="center" valign="middle"><?php echo $res[$i]['database_name'];?></td>
                <td align="center" valign="middle"><?php echo $res[$i]['dates'];?></td>
                <td align="center" valign="middle"><a href="#">删除</a></td>
          </tr>
<?php
    }
} catch (PDOException $e) {
    die ("Error!: " . $e->getMessage() . "<br/>");
} ?>
```

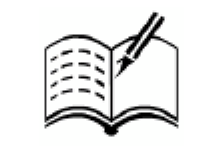
运行结果如图 14.11 所示。





ID	PDO	数据库	时间	操作
1	pdo	mysql	2010-09-18 10:19:17	删除
2	pdo	Oracle	2010-09-18 10:19:17	删除
4	pdo	Access	2010-10-10 00:00:00	删除
22	pdo	SQL	2010-10-09 10:59:30	删除
23	pdo	db2	2010-10-16 10:59:41	删除

图 14.11　fetchAll 方法返回结果集中的所有数据



14.4.3　fetchColumn 方法

fetchColumn 方法获取结果集中下一行指定列的值。其语法如下：

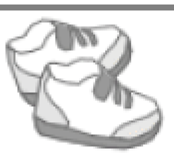
```
string PDOStatement::fetchColumn ( [int column_number] )
```

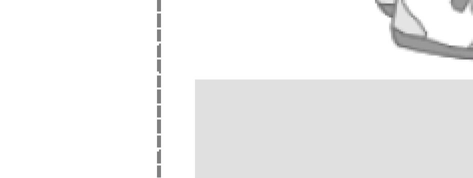
可选参数 column_number 设置行中列的索引值，该值从 0 开始。如果省略该参数则将从第 1 列开始取值。

通过 fecthColumn 方法获取结果集中下一行中指定列的值，注意这里是“结果集中下一行中指定列的值”。本实例输出数据表中第一列的值，即输出数据的 ID。具体步骤如下：

例 14.6　创建 index.php 文件，设计网页页面。首先，通过 PDO 连接 MySQL 数据库。然后，定义 SELECT 查询语句，应用 prepare 和 execute 方法执行查询操作。接着，通过 fetchColumn 方法输出结果集中下一行第一列的值。其关键代码如下：（**实例位置：配套资源\mr\14\ example\14.6**）

```
<?php
$dbms='mysql';          //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，而无须记住那么多的函数
$host='localhost';                  //数据库主机名
$dbName='db_database14';            //使用的数据库
$user='root';                       //数据库连接用户名
$pass='111';                        //对应的密码
$dsn="$dbms:host=$host;dbname=$dbName";
try {
    $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
    $query="select * from tb_pdo_mysql";        //定义SQL语句
    $result=$pdo->prepare($query);              //准备查询语句
    $result->execute();                         //执行查询语句，并返回结果集
?>
        <tr>
          <td height="22" align="center" valign="middle"><?php echo $result->fetchColumn(0); ?></td>
        </tr>
        <tr>
          <td height="22" align="center" valign="middle"><?php echo $result->fetchColumn(0); ?></td>
        </tr>
        <tr>
          <td height="22" align="center" valign="middle"><?php echo $result->fetchColumn(0); ?></td>
```





```
                </tr>
                <tr>
                  <td height="22" align="center" valign="middle"><?php echo $result->fetchColumn(0);
?></td>
                </tr>
        <?php
             } catch (PDOException $e) {
            die ("Error!: " . $e->getMessage() . "<br/>");
        }
             ?>
```

运行结果如图 14.12 所示。

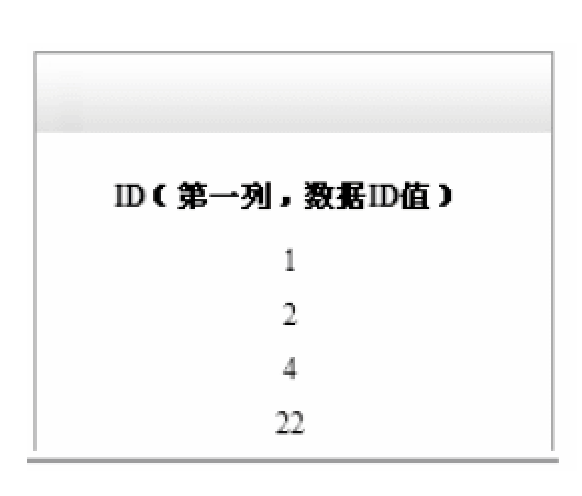


图 14.12　fetchColumn 方法获取结果集中第一列的值

上机演练

上机演练 7　浏览客户留言

本例实现浏览数据库中存储的客户留言信息的功能。其利用 SQL 语句的排序方法，根据 ID 进行降幂排列，应用 fetch 方法循环输出数据表中的数据。其运行效果如图 14.13 所示。

浏览客户留言

ID	标题	内容	日期
9	你好编程宝典	编程宝典是您最好的选择	2010-11-24
8	你好编程宝典	编程宝典是您最好的选择	2010-11-24
7	你好编程宝典	编程宝典是您最好的选择	2010-11-24
6	你好编程宝典	编程宝典是您最好的选择	2010-11-24

图 14.13　浏览客户留言

14.5　PDO 中捕获 SQL 语句中的错误

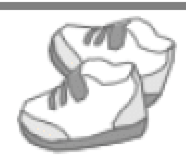
视频讲解：配套资源\mr\14\video\PDO 中捕获 SQL 语句中的错误.exe

在 PDO 中捕获 SQL 语句中的错误有 3 种方案可以选择。

14.5.1　使用默认模式——PDO::ERRMODE_SILENT

在默认模式中设置 PDOStatement 对象的 errorCode 属性，但不进行其他任何操作。

通过 prepare 和 execute 方法向数据库中添加数据，设置 PDOStatement 对象的 errorCode 属性，手动检测代码中的错误，具体步骤如下：





例 14.7　创建 index.php 文件，添加 form 表单，将表单元素提交到本页。通过 PDO 连接 MySQL 数据库，通过预处理语句 prepare 和 execute 执行 INSERT 添加语句，向数据表中添加数据，并且设置 PDOStatement 对象的 errorCode 属性，检测代码中的错误。其关键代码如下：**（实例位置：配套资源\mr\14\example\14.7）**

```
<?php
if($_POST['Submit']=="提交" && $_POST['pdo']!=""){
    $dbms='mysql';                          //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，而无须记住那么多的函数
    $host='localhost';                      //数据库主机名
    $dbName='db_database14';                //使用的数据库
    $user='root';                           //数据库连接用户名
    $pass='111';                            //对应的密码
    $dsn="$dbms:host=$host;dbname=$dbName";
        $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
    $query="insert into tb_pdo_mysqls(pdo_type,database_name,dates)values('".$_POST['pdo']."','".$_POST['databases']."','".$_POST['dates']."')";
    $result=$pdo->prepare($query);
    $result->execute();
    $code=$result->errorCode();
    if(empty($code)){
        echo "数据添加成功！ ";
    }else{
        echo '数据库错误：<br/>';
        echo 'SQL Query:'.$query;
        echo    '<pre>';
        var_dump($result->errorInfo());
        echo '</pre>';
    }
}
?>
```

在本实例中，在定义 INSERT 添加语句时使用了错误的数据表名称 tb_pdo_mysqls（正确名称是 tb_pdo_mysql），导致输出的结果如图 14.14 所示。

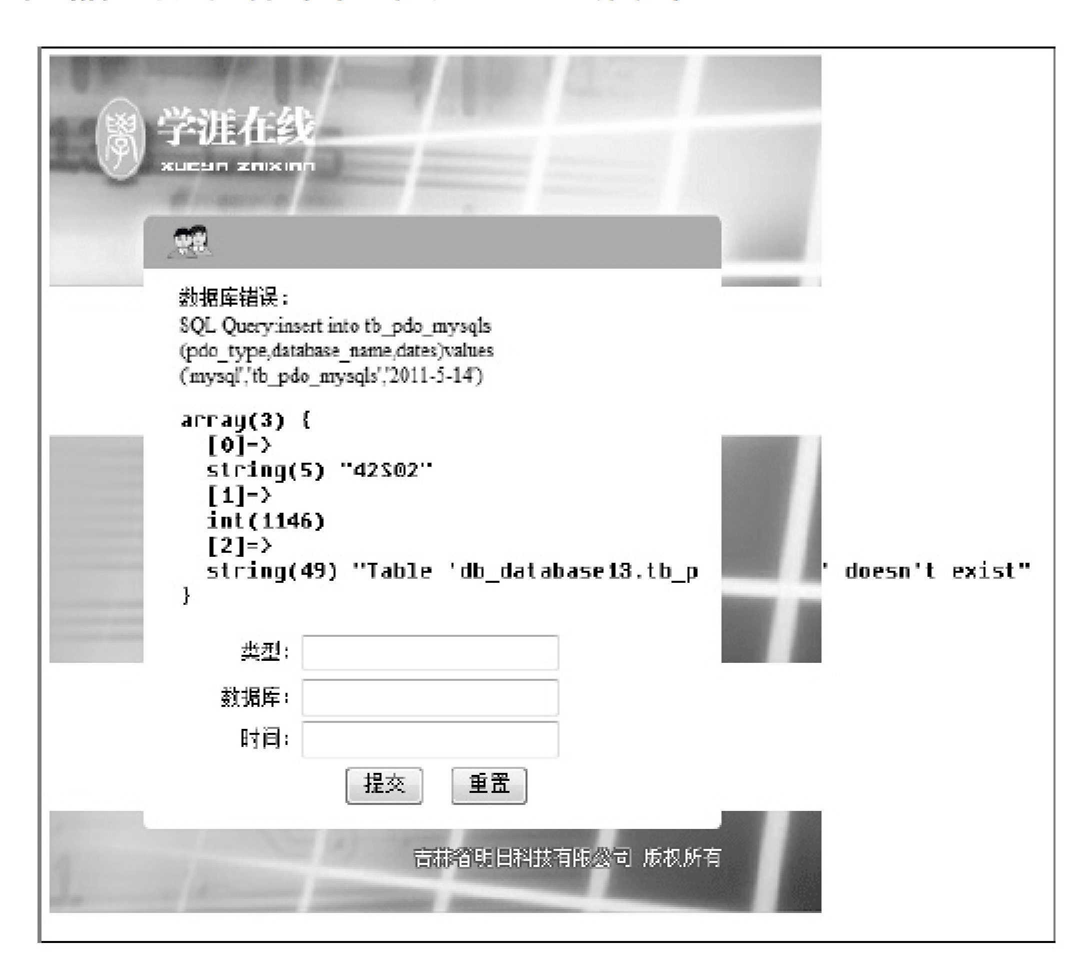


图 14.14　在默认模式中捕获 SQL 中的错误





Note

14.5.2 使用警告模式——PDO::ERRMODE_WARNING

警告模式会产生一个 PHP 警告，并设置 errorCode 属性。如果设置的是警告模式，那么除非明确地检查错误代码，否则程序将继续按照其方式运行。

设置警告模式，通过 prepare 和 execute 方法读取数据库中数据，并且通过 while 语句和 fetch 方法完成数据的循环输出，体会在设置成警告模式后执行错误的 SQL 语句。具体步骤如下：

例 14.8 创建 index.php 文件，连接 MySQL 数据库，通过预处理语句 prepare 和 execute 执行 SELECT 查询语句，并设置一个错误的数据表名称，同时通过 setAttribute 方法设置为警告模式，最后通过 while 语句和 fetch()方法完成数据的循环输出。其关键代码如下：（**实例位置：配套资源\mr\14\example\14.8**）

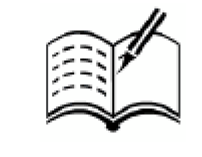
```
<?php
$dbms='mysql';          //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，
而无须记住那么多的函数
$host='localhost';                   //数据库主机名
$dbName='db_database14';             //使用的数据库
$user='root';                        //数据库连接用户名
$pass='111';                         //对应的密码
$dsn="$dbms:host=$host;dbname=$dbName";
try {
    $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_WARNING);//设置为警告模式
    $query="select * from tb_pdo_mysqls";    //定义SQL语句
    $result=$pdo->prepare($query);           //准备查询语句
    $result->execute();                      //执行查询语句，并返回结果集
    while($res=$result->fetch(PDO::FETCH_ASSOC)){        //while循环输出查询结果集，并且
设置结果集的为关联索引
    ?>
          <tr>
            <td height="22" align="center" valign="middle"><?php echo $res['id'];?></td>
            <td align="center" valign="middle"><?php echo $res['pdo_type'];?></td>
            <td align="center" valign="middle"><?php echo $res['database_name'];?></td>
            <td align="center" valign="middle"><?php echo $res['dates'];?></td>
          </tr>
<?php
    }
      } catch (PDOException $e) {
    die ("Error!: " . $e->getMessage() . "<br/>");
}
      ?>
```

在设置为警告模式后，如果 SQL 语句出现错误，将给出一个提示信息，但是程序仍能够继续执行下去，其运行结果如图 14.15 所示。





图 14.15　设置警告模式后捕获的 SQL 语句错误



14.5.3　使用异常模式——PDO::ERRMODE_EXCEPTION

异常模式会创建一个 PDOException，并设置 errorCode 属性。它可以将执行代码封装到一个 try{…}catch{…}语句块中。未捕获的异常将会导致脚本中断，并显示堆栈跟踪让用户了解是哪里出现的问题。

例 14.9　在执行数据库中数据的删除操作时，设置为异常模式，并且编写一个错误的 SQL 语句（操作错误的数据表 tb_pdo_mysqls），体会异常模式、警告模式以及默认模式的区别。具体步骤如下。**（实例位置：配套资源\mr\14\example\14.9）**

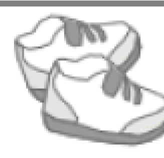
（1）创建 index.php 文件，连接 MySQL 数据库，通过预处理语句 prepare 和 execute 执行 SELECT 查询语句，通过 while 语句和 fetch 方法完成数据的循环输出，并且设置删除超链接，链接 delete.php 文件，传递的参数是数据的 ID 值。其运行效果如图 14.16 所示。



图 14.16　数据的循环输出

（2）创建 delete.php 文件，获取超链接传递的数据 ID 值，连接数据库，通过 setAttribute 方法设置为异常模式，定义 DELETE 删除语句，删除一个错误数据表（tb_pdo_mysqls）中的数据，并且通过 try{…}catch{…}语句捕获错误信息。其代码如下：

```
<?php
header ( "Content-type: text/html; charset=utf-8" );    //设置文件编码格式
if($_GET['conn_id']!=""){
    $dbms='mysql';        //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，
无须记住那么多的函数
    $host='localhost';                          //数据库主机名
    $dbName='db_database14';                    //使用的数据库
    $user='root';                               //数据库连接用户名
    $pass='111';                                //对应的密码
    $dsn="$dbms:host=$host;dbname=$dbName";
    try {
        $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
        $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
        $query="delete from tb_pdo_mysqls where Id=:id";
        $result=$pdo->prepare($query);          //预准备语句
        $result->bindParam(':id',$_GET['conn_id']);    //绑定更新的数据
```

```
            $result->execute();
        } catch (PDOException $e) {
            echo 'PDO Exception Caught.';
            echo 'Error with the database:<br/>';
            echo   'SQL Query: '.$query;
            echo '<pre>';
            echo "Error: " . $e->getMessage(). "<br/>";
            echo "Code: " . $e->getCode(). "<br/>";
            echo "File: " . $e->getFile(). "<br/>";
            echo "Line: " . $e->getLine(). "<br/>";
            echo "Trace: " . $e->getTraceAsString(). "<br/>";
            echo '</pre>';
        }
    }
    ?>
```

在设置为异常模式后，执行错误的 SQL 语句返回的结果如图 14.17 所示。

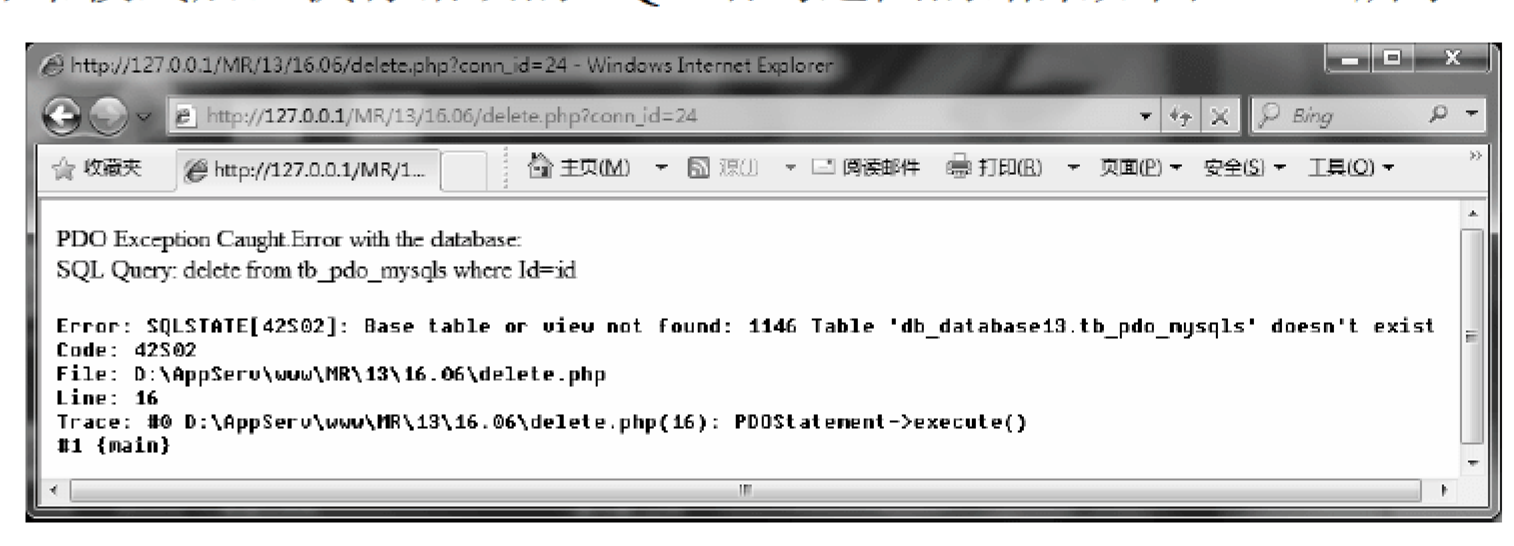


图 14.17　异常模式捕获的 SQL 语句错误信息

14.6　PDO 中错误处理

视频讲解：配套资源\mr\14\video\PDO 中错误处理.exe

在 PDO 中有两个获取程序中错误信息的方法：errorCode 方法和 errorInfo 方法。

14.6.1　errorCode 方法

errorCode 方法用于获取在操作数据库句柄时所发生的错误代码，这些错误代码被称为 SQLSTATE 代码。其语法格式如下：

```
int PDOStatement::errorCode ( void )
```

errorCode 方法返回一个 SQLSTATE，它是由 5 个数字和字母组成的代码。

例 14.10　在 PDO 中通过 query 方法完成数据的查询操作，并且通过 foreach 语句完成数据的循环输出。在定义 SQL 语句时使用一个错误的数据表，并且通过 errorCode 方法返回错误代码。具体实现如下：（**实例位置：配套资源\mr\14\example\14.10**）

创建 index.php 文件。首先通过 PDO 连接 MySQL 数据库。然后通过 query 方法执行查询语句。接着通过 errorCode 方法获取错误代码。最后通过 foreach 语句完成数据的循环输出。其关键代码如下：

```
    <?php
$dbms='mysql';          //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，
```





```
而无须记住那么多的函数
        $host='localhost';                  //数据库主机名
        $dbName='db_database14';            //使用的数据库
        $user='root';                       //数据库连接用户名
        $pass='111';                        //对应的密码
        $dsn="$dbms:host=$host;dbname=$dbName";
        try {
        $pdo = new PDO($dsn, $user, $pass);     //初始化一个PDO对象，即创建了数据库连接对象$pdo
        $query="select * from tb_pdo_mysqls";   //定义SQL语句
        $result=$pdo->query($query);            //执行查询语句，并返回结果集
        echo "errorCode为：".$pdo->errorCode();
        foreach($result as $items){
            ?>
                <tr>
                  <td height="22" align="center" valign="middle"><?php echo $items['id'];?></td>
                  <td align="center" valign="middle"><?php echo $items['pdo_type'];?></td>
                  <td align="center" valign="middle"><?php echo $items['database_name'];?></td>
                  <td align="center" valign="middle"><?php echo $items['dates'];?></td>
                </tr>
                <?php
                }
            } catch (PDOException $e) {
            die ("Error!: " . $e->getMessage() . "<br/>");
        }
            ?>
```

运行结果如图 14.18 所示。



图 14.18　通过 errorCode 方法获取错误代码

14.6.2　errorInfo 方法

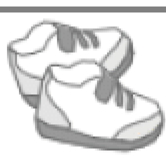
errorInfo 方法用于获取操作数据库句柄时所发生的错误信息。其语法格式如下：

```
array PDOStatement::errorInfo ( void )
```

errorInfo 方法的返回值为一个数组，它包含了相关的错误信息。

例 14.11　在 PDO 中通过 query 方法完成数据的查询操作，并且通过 foreach 语句完成数据的循环输出。在定义 SQL 语句时使用一个错误的数据表，并且通过 errorInfo 方法返回错误信息。（**实例位置：配套资源\mr\14\example\14.11**）

创建 index.php 文件。首先通过 PDO 连接 MySQL 数据库。然后通过 query 方法执行查询语句。接着通过 errorInfo 方法获取错误信息。最后通过 foreach 语句完成数据的循环输出。其关键代码如下：



```
<?php
$dbms='mysql';     //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，无须记住那么多的函数
$host='localhost';                    //数据库主机名
$dbName='db_database14';              //使用的数据库
$user='root';                         //数据库连接用户名
$pass='111';                          //对应的密码
$dsn="$dbms:host=$host;dbname=$dbName";
try {
$pdo = new PDO($dsn, $user, $pass);   //初始化一个PDO对象，即创建了数据库连接对象$pdo
    $query="select * from tb_pdo_mysqls";   //定义SQL语句
    $result=$pdo->query($query);            //执行查询语句，并返回结果集
    print_r($pdo->errorInfo());
 foreach($result as $items){
 ?>
        <tr>
          <td height="22" align="center" valign="middle"><?php echo $items['id'];?></td>
          <td align="center" valign="middle"><?php echo $items['pdo_type'];?></td>
          <td align="center" valign="middle"><?php echo $items['database_name'];?></td>
          <td align="center" valign="middle"><?php echo $items['dates'];?></td>
        </tr>
        <?php
        }
        } catch (PDOException $e) {
    die ("Error!: " . $e->getMessage() . "<br/>");
}
     ?>
```

运行结果如图 14.19 所示。

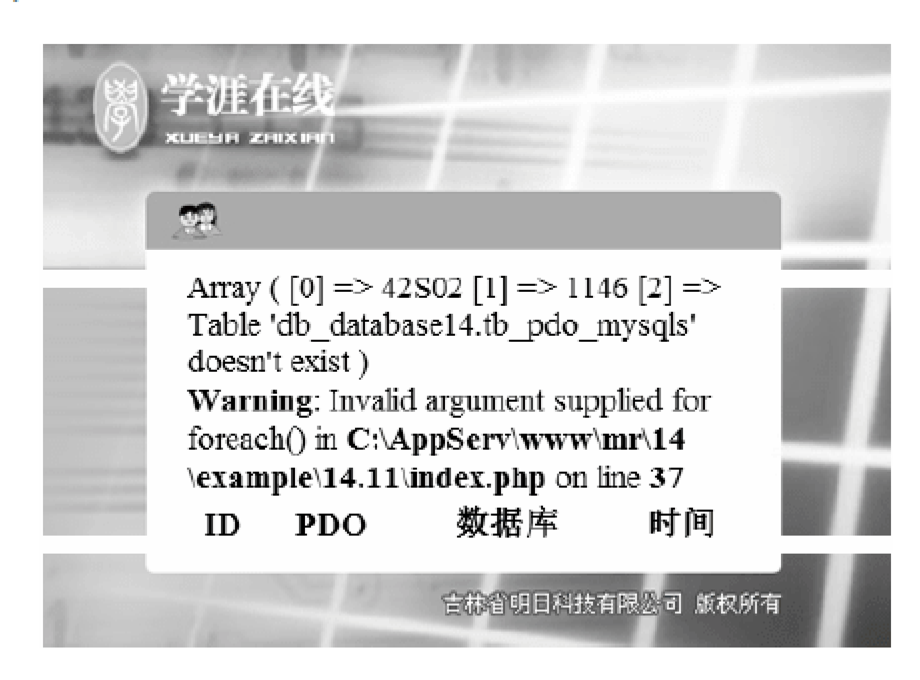


图 14.19　通过 errorInfo 方法获取错误信息

14.7　PDO 中事务处理

在 PDO 中同样可以实现事务处理的功能，其应用的方法如下：

☑　开启事务——beginTransaction 方法

beginTransaction 方法将关闭自动提交（autocommit）模式，直到事务提交或回滚以后才



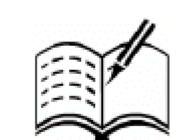
恢复。

☑　提交事务——commit 方法

commit()方法完成事务的提交操作，若成功则返回 True，否则返回 False。

☑　事务回滚——rollback 方法

rollback()方法执行事务的回滚操作。



例 14.12　通过 prepare 和 execute 方法向数据库中添加数据，并且通过事务处理机制确保数据能够正确地添加到数据中。具体实现如下：（**实例位置：配套资源\mr\14\example\14.12**）

创建 index.php 文件。首先，定义数据库连接的参数，创建 try{…}catch{…}语句，在 try{…}语句中实例化 PDO 构造函数，完成与数据库的连接，并且通过 beginTransaction 方法开启事务。然后，定义 INSERT 添加语句，通过$_POST[]方法获取表单中提交的数据，通过 prepare 和 execute 方法向数据库中添加数据，并且通过 commit 方法完成事务的提交操作。最后，在 catch{…}语句中返回错误信息，并且通过 rollBack 执行事务的回滚操作。其代码如下：

```
<?php
if($_POST['Submit']=="提交" && $_POST['pdo']!=""){
    $dbms='mysql';                    //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，无须记住那么多的函数
    $host='localhost';                //数据库主机名
    $dbName='db_database14';          //使用的数据库
    $user='root';                     //数据库连接用户名
    $pass='111';                      //对应的密码
    $dsn="$dbms:host=$host;dbname=$dbName";
    try {
        $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
        $pdo->beginTransaction();           //开启事务
        $query="insert into tb_pdo_mysql(pdo_type,database_name,dates)values('".$_POST['pdo']."','".$_POST['databases']."','".$_POST['dates']."')";
        $result=$pdo->prepare($query);
        if($result->execute()){
            echo "数据添加成功！";
        }else{
            echo "数据添加失败！";
        }
        $pdo->commit();                     //执行事务的提交操作
    } catch (PDOException $e) {
        die ("Error!: " . $e->getMessage() . "<br/>");
        $pdo->rollBack();                   //执行事务的回滚
    }
}
?>
```

运行结果如图 14.20 所示。



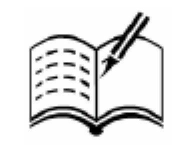



图 14.20　数据添加中应用事务处理机制

14.8　PDO 中存储过程

存储过程允许在更接近于数据的位置操作数据，从而减少带宽的使用，它们使数据独立于脚本逻辑，允许使用不同语言的多个系统以相同的方式访问数据，从而节省了花费在编码和调试上的宝贵时间。同时它使用预定义的方案执行操作，提高查询速度，并且能够阻止与数据的直接相互作用，从而起到保护数据的作用。

下面讲解如何在 PDO 中调用存储过程。这里首先创建一个存储过程，其 SQL 语句如下：

```
drop procedure if exists pro_reg;
delimiter //
create procedure pro_reg (in nc varchar(80), in pwd varchar(80), in email varchar(80),in address varchar(50))
begin
insert into tb_reg (name, pwd ,email ,address) values (nc, pwd, email, address);
end;
//
```

“drop”语句删除 MySQL 服务器中已经存在的存储过程 pro_reg。

“delimiter //”的作用是将语句结束符更改为“//”。

“in nc varchar(50)……in address varchar(50)”表示要向存储过程中传入的参数。

“begin……end”表示存储过程中的语句块，它的作用类似于 PHP 语言中的“{……}”。

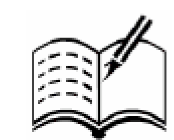
在存储过程创建成功后，下面调用这个存储过程实现用户注册的功能。

例 14.13　在 PDO 中通过 call 语句调用存储过程，实现用户注册信息的添加操作。具体实现如下：**（实例位置：配套资源\mr\14\example\14.13）**

创建 index.php 文件。首先，创建 form 表单，将用户注册信息通过 POST 方法提交到本页。然后，在本页中编写 PHP 脚本，通过 PDO 连接 MySQL 数据库，并且设置数据库编码格式为 UTF-8，获取表单中提交的用户注册信息。接着，通过 call 语句调用存储过程 pro_reg，将用户注册信息添加到数据表中。最后，通过 try{…}catch{…}语句块返回错误信息。其关键代码如下：

```
<?php
 if($_POST['submit']!=""){
 $dbms='mysql';     //数据库类型，对于开发者来说，使用不同的数据库，只要修改它即可，无
须记住那么多的函数
```



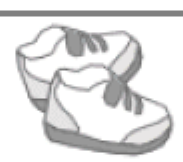
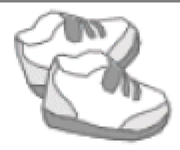


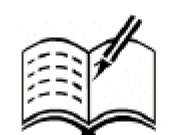
```
        $host='localhost';                  //数据库主机名
        $dbName='db_database14';            //使用的数据库
        $user='root';                       //数据库连接用户名
        $pass='111';                        //对应的密码
        $dsn="$dbms:host=$host;dbname=$dbName";
        try {
            $pdo = new PDO($dsn, $user, $pass); //初始化一个PDO对象，即创建了数据库连接对象$pdo
            $pdo->query("set names utf8"); //设置数据库编码格式
            $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);    //定义
错误异常模式
         $nc=$_POST['nc'];
            $pwd=md5($_POST['pwd']);
            $email=$_POST['email'];
            $address=$_POST['address'];
            $query="call pro_reg('$nc','$pwd','$email','$address')";
            $result=$pdo->prepare($query);
            if($result->execute()){
                echo "数据添加成功！ ";
            }else{
                echo "数据添加失败！ ";
            }
        } catch (PDOException $e) {
            echo 'PDO Exception Caught.';
            echo 'Error with the database:<br/>';
            echo   'SQL Query: '.$query;
            echo '<pre>';
            echo "Error: " . $e->getMessage(). "<br/>";
            echo "Code: " . $e->getCode(). "<br/>";
            echo "File: " . $e->getFile(). "<br/>";
            echo "Line: " . $e->getLine(). "<br/>";
            echo "Trace: " . $e->getTraceAsString(). "<br/>";
            echo '</pre>';
        }
    }
    ?>
```

运行结果如图 14.21 所示。



图 14.21　通过存储过程完成用户的注册



本 章 摘 要

1．如何通过 PDO 连接不同的数据库。
2．如何应用 PDO 中的 exec、query 和预处理语句执行 SQL 语句。
3．如何通过 fetch、fetchAll 或 fetchColumn 方法获取结果集和错误处理。
4．PDO 的高级应用事务和存储过程。

习　　题

1．下列有关 PDO 的索引是关联索引的函数是（　　）。

A．PDO::LOWER　B．PDO::ASSOC　C．PDO::FETCH_NUM

2. 如果在 PHP 中使用 Oracle 作为数据库服务器，应该在 PDO 中加载下面哪个驱动程序？（　　）

A．PDO_DBLIB　B．PDO_MYSQL　C．PDO_OCI　D．PDO_ODBC

3．当 PDO 对象创建成功以后，与数据库的连接即已经建立，就可以使用 PDO 对象了。下面哪个不是 PDO 对象中的成员方法？（　　）

A．phpInfo　B．query　C．exec　D．prepare

4．PDO 提供了多种不同的错误处理模式，不仅可以满足不同风格的编程，而且可以调整扩展处理错误的方式。下面哪个不是 PDO 提供的错误处理模式？（　　）

A．ERRMODE_SILENT　B．ERRMODE_WARNING
C．PDO::ERRMODE_ERROR　D．ERRMODE_EXCEPTION

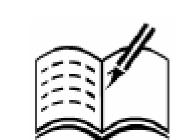
5．下面哪个函数属于 PDO 预处理语句？（　　）

A．PDO::ATTR_AUTOCOMMIT　B．PREPARE
C．PDO::ERRMODE_ERROR　D．ERRMODE_EXCEPTION

6．开启 PDO 错误默认模式代码为（　　）。
7．开启 PDO 错误警告模式代码为（　　）。
8．开启 PDO 错误异常模式代码为（　　）。
9．执行 PDO 插入操作需要使用（　　）函数。
10．假设 MySQL 数据库名称为 db_database14，数据库用户名为 root，密码为 111，数据源为 localhost，程序运行后页面中显示的内容为（　　）。

```
header("Content-Type:text/html;charset=utf-8");
$dbms='mysql';
$dbName='db_database14';
$user='root';
$pwd='111';
$host='localhost';
$dsn="$dbms:host=$host;dbname=$dbName";
try {
    $pdo=new PDO($dsn,$user,$pwd);
```






```
        echo "PDO连接MySQL成功";
    } catch (Exception $e) {
        echo $e->getMessage()."<br>";
    }
```

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 13 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　明日书店会员注册

注册系统在本书的很多章节中都有体现，本次所讲解的是利用 PDO 抽象层实现的会员注册系统。其核心思想是拼接插入的 SQL 代码，将文本框的相关信息动态地添加到数据表中。运行效果如图 14.22 所示。



图 14.22　会员注册系统

实战模拟 2　添加留言信息

添加留言信息是非常普通和常见的模块操作，从原理上说其实它与注册登录操作几乎一样，笔者意在让读者熟悉 PDO 操作常见的几种模式，使读者熟能生巧。添加留言信息的运行效果如图 14.23 所示。

实战模拟 3　查询留言

实现一个站内搜索的功能，根据提交的关键字查询出符合条件的数据信息。其运行效果如图 14.24 所示。

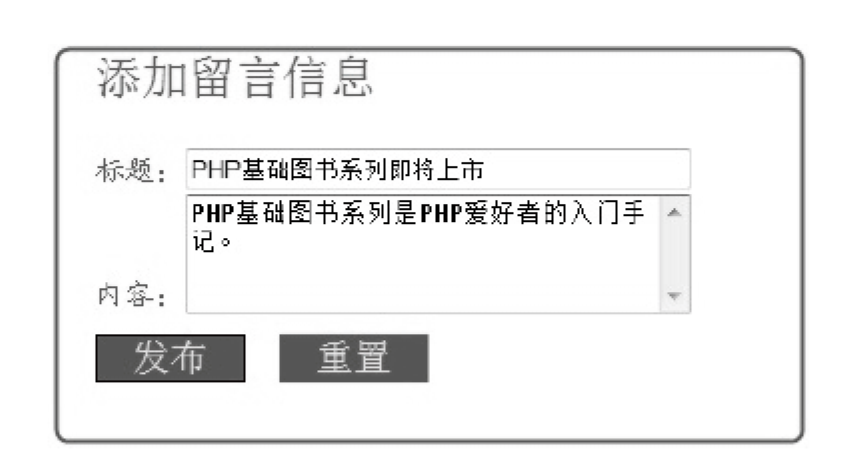


图 14.23　添加留言信息

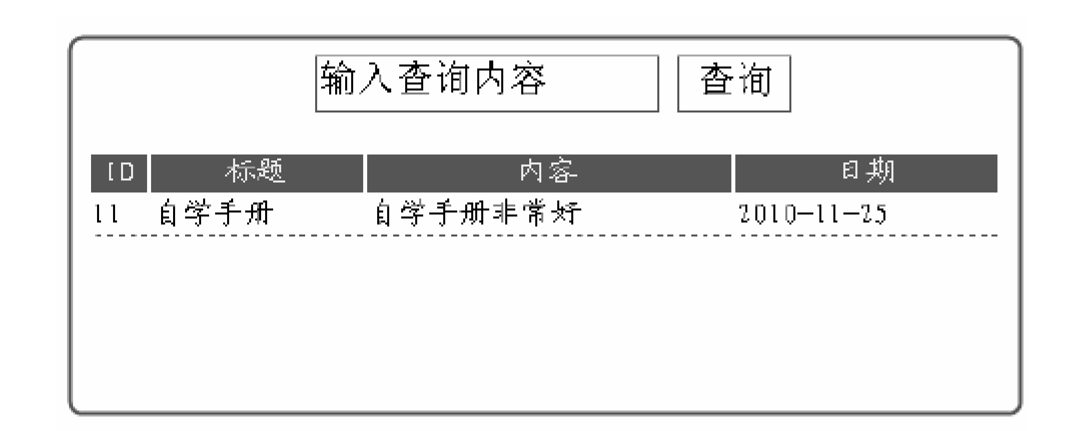


图 14.24　查询留言内容



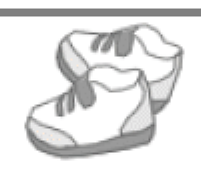
Smarty 模板

（ 自学视频、源程序：配套资源\mr\15\ ）

Smarty 模板引擎实现 Web 网站中用户界面和 PHP 代码的分离。通过 Smarty 模板的加入，将尝试一种新的 Web 交互式网站的开发流程。这种开发流程避免了传统开发流程中分工不明确，如美工要懂 PHP 代码、程序员要会网页设计等问题。有了 Smarty 模板的支持，美工不再需要面对错综复杂的 PHP 代码，程序员也不再需要进行枯燥的抠图工作，可以做到各司其职，各尽其责。

学习摘要：

- 走进 Smarty 模板引擎
- Smarty 模板的安装和配置
- 基础的 Smarty 语法
- Smarty 模板设计变量
- 变量调节器
- 内建函数（foreach、include、if 等语句）
- 自定义函数
- 配置文件
- Smarty 常量
- Smarty 程序设计变量
- Smarty 方法
- Smarty 缓存





15.1　走进 Smarty 模板引擎

视频讲解：配套资源\mr\15\video\走进 Smarty 模板引擎.exe

Smarty 是一个使用 PHP 编写的 PHP 模板引擎，是目前业界最著名、功能最强大的一种 PHP 模板引擎。它将一个应用程序分成两部分：视图和逻辑控制，也就是将 UI（用户界面）和 PHP code（PHP 代码）分离。这样，程序员在修改程序时就不会影响页面设计，而美工在重新设计或修改页面时也不会影响程序逻辑。Smarty 模板引擎的运行流程如图 15.1 所示。

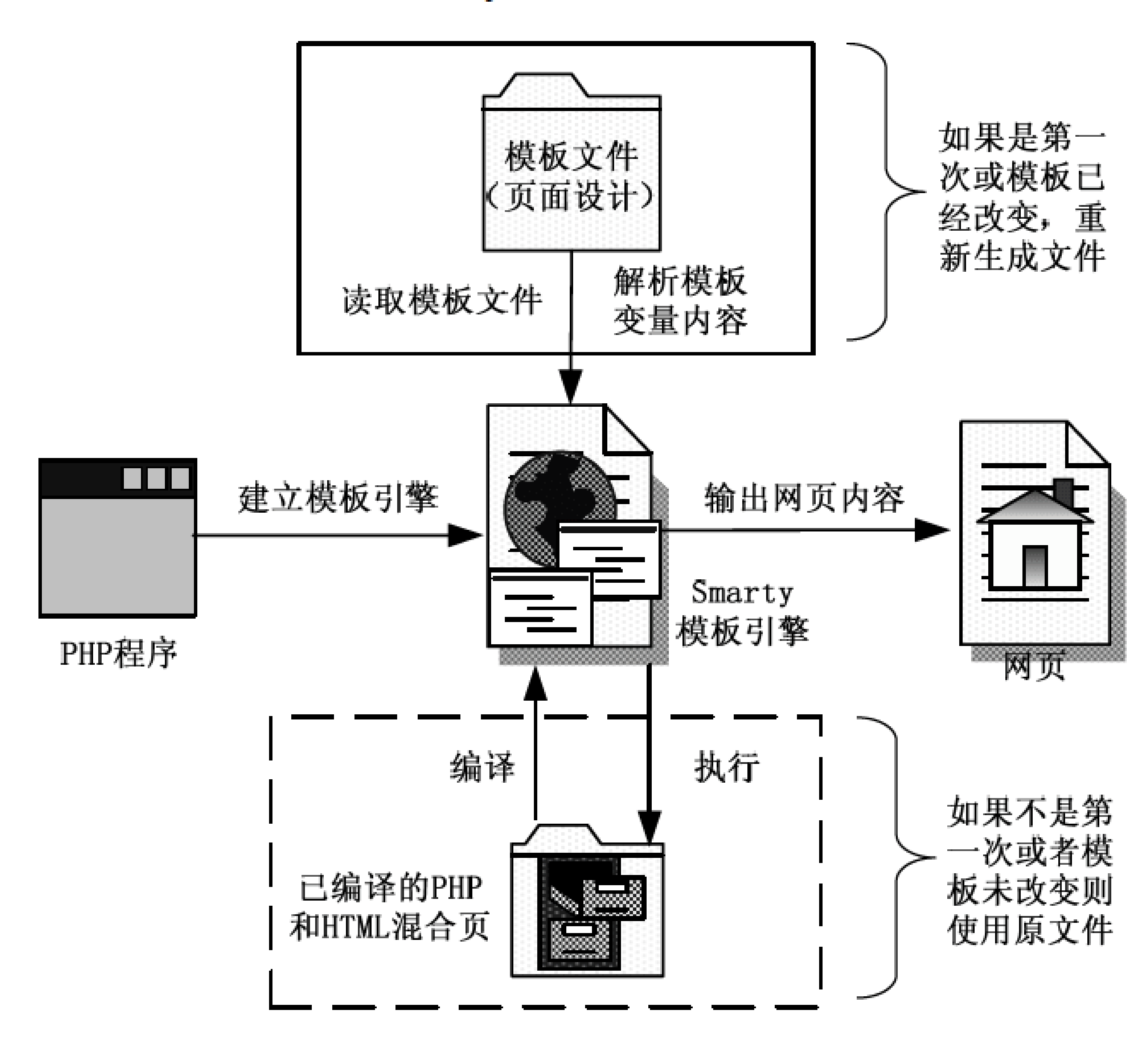


图 15.1　Smarty 模板引擎的运行流程

Smarty 拥有丰富的函数库，从统计字数到字符串的截取、文字的环绕以及正则表达式都可以直接使用，还具有很强的扩展能力。Smarty 模板的优点总结如下：

☑ 速度：相对于其他模板而言，采用 Smarty 模板编写的程序可以获得最快的速度。

☑ 编译性调用：采用 Smarty 模板编写的程序在运行时会生成一个 PHP 和 HTML 混合的文件，在下一次访问模板时会直接访问这个混编的文件（前提是源文件没有发生改变），而不必重新编译，进而可以提高访问速度。

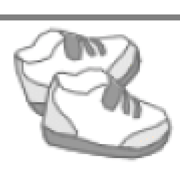
☑ 缓存技术：Smarty 提供一种可选择的缓存技术，可以将客户端的 HTML 文件缓存成一个静态页。当用户开启缓存后，在指定的时间内，Web 请求会直接调用这个缓存文件，即直接调用静态的 HTML 文件。

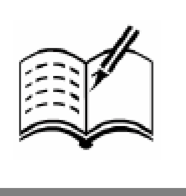
☑ 插件技术：因为 Smarty 模板引擎是通过 PHP 面向对象技术实现的，所以不仅可以修改 Smarty 模板的源文件，而且可以通过自定义函数向 Smarty 中添加功能。

☑ 模板中可以使用 if/elseif/else/endif。

15.1.1　Smarty 模板引擎下载

PHP 没有内置 Smarty 模板类，需要单独下载和配置，同时 Smarty 要求服务器上的 PHP 版





Note

本最低为 4.0.6。用户可以从 http://smarty.net/download.php 处下载最新的 Smarty 压缩包。本书中使用的版本是 Smarty-2.6.23。Smarty 模板下载页面如图 15.2 所示。

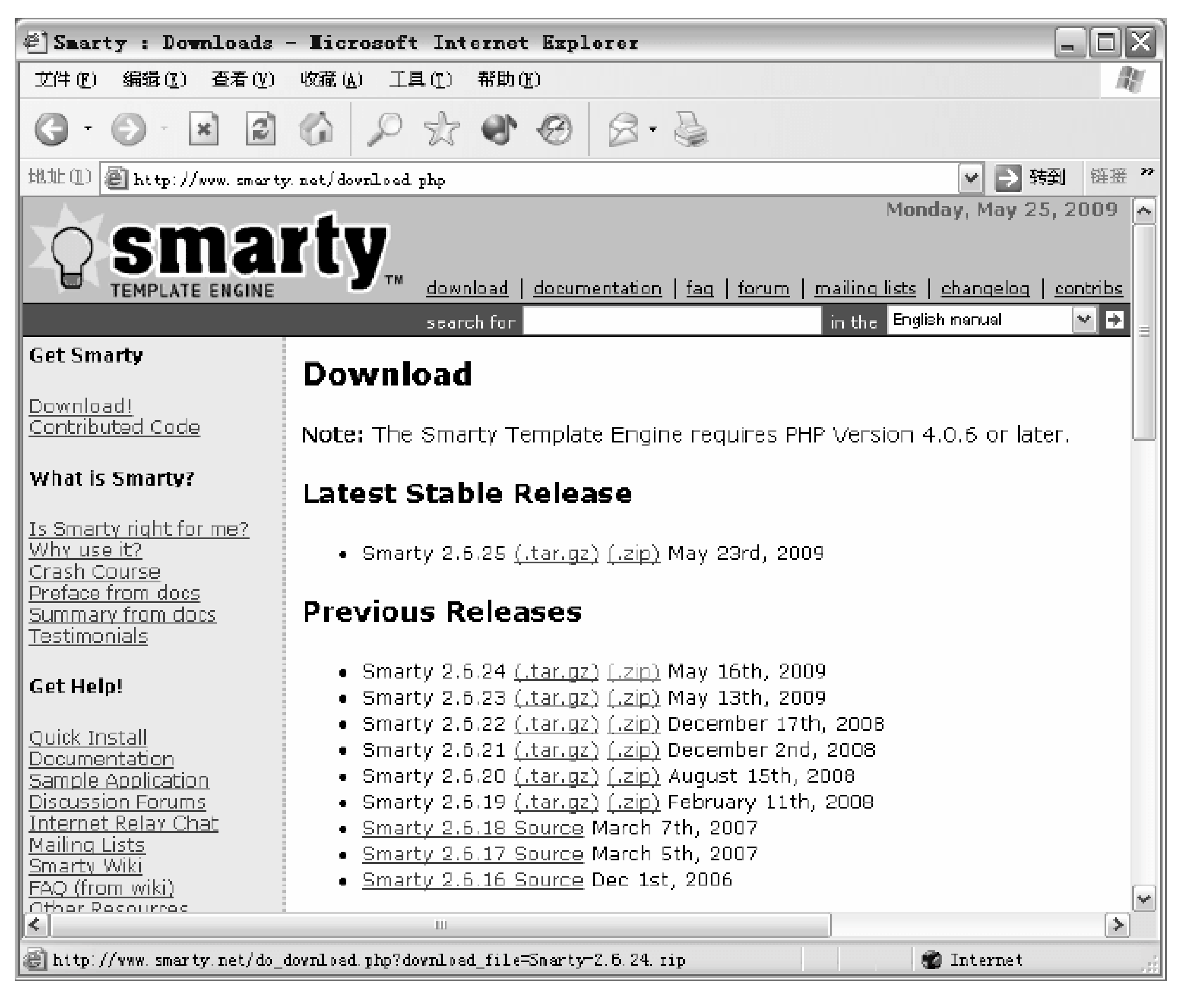


图 15.2 Smarty 模板下载页面

15.1.2 Smarty 模板引擎安装

（1）将下载的 Smarty 压缩包解压后，有一个 libs 目录，这里包含了 Smarty 类库的核心文件，包括 smarty.class.php、smarty_Compiler.class.php、config_File.class.php 和 debug.tpl 4 个文件，另外还有 internals 和 plug-ins 两个目录，如图 15.3 所示。

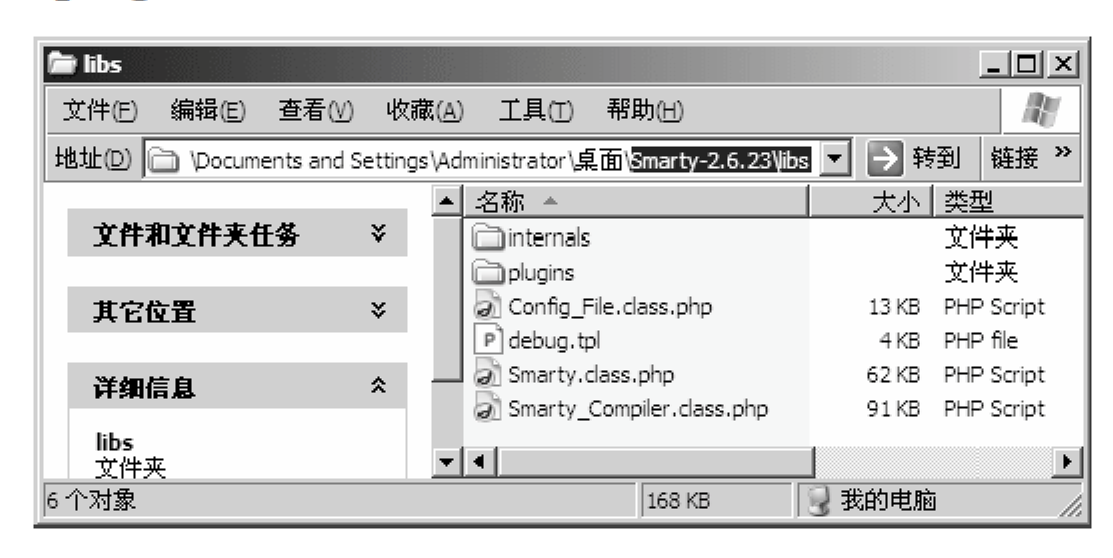


图 15.3 Smarty-2.6.23 文件夹

（2）复制 libs 目录到服务器根目录指定的文件夹下。至此 Smarty 安装成功。

15.1.3 Smarty 模板引擎配置

Smarty 模板引擎的配置步骤如下：

（1）确定 Smarty 类库的存储位置。因为 Smarty 类库是通用的，每一个项目都可能会使用到它，所以将 Smarty 存储在根目录下是一个比较理想的选择。这里以将其存储于本章根目录 15\example\15.1\Smarty\文件夹下为例（为第一个 Smarty 程序配置环境）。

（2）新建 4 个目录 templates、templates_c、configs 和 cache，并将它们存储于 15\example\15.1\Smarty 文件夹下，如图 15.4 所示。





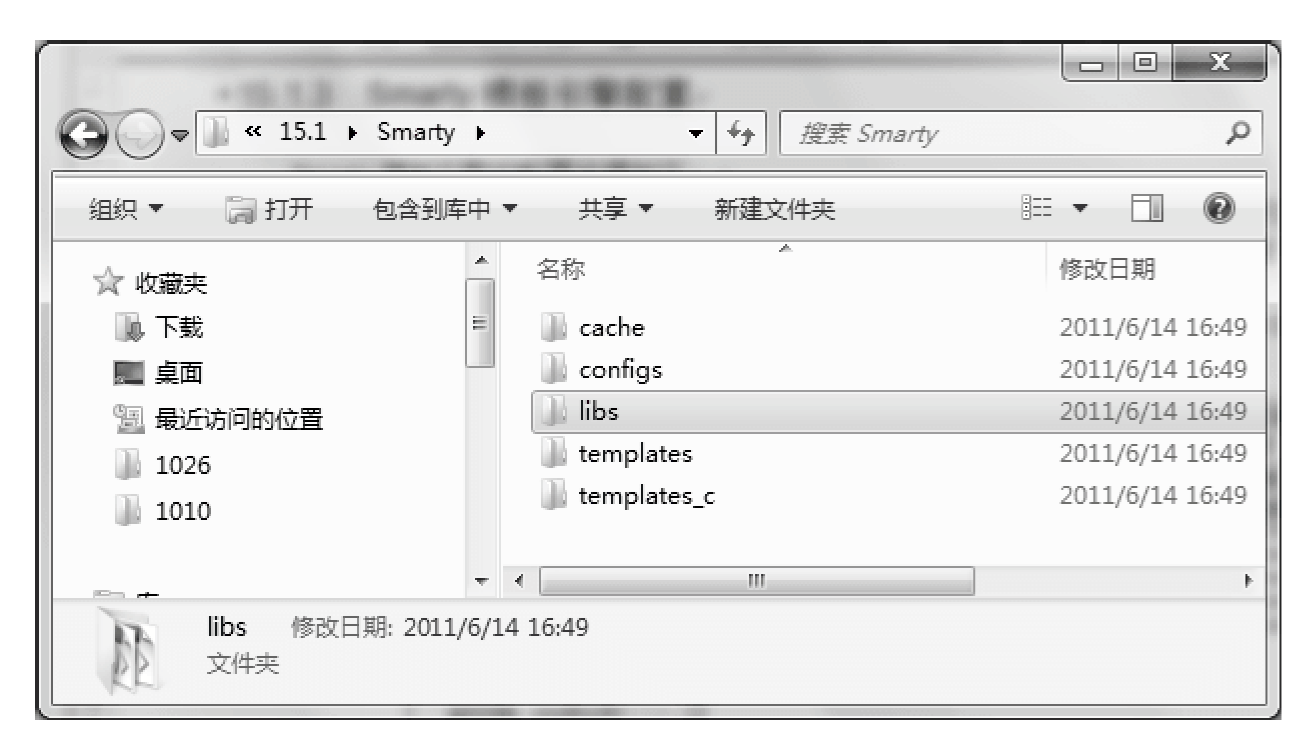


图 15.4 Smarty 目录中的内容

指点迷津：

新建的 4 个目录并没有固定的存储位置，只要配置的路径正确，则存储在根目录下的任何位置都可以。

（3）创建配置文件。当所有需要的资源都就绪之后，下面要做的就是将它们进行整合并调动起来。这就是配置文件的作用，定义服务器、Smarty 的绝对路径；加载 Smarty 类库文件；为创建的 4 个目录设置正确的路径；定义 Smarty 的定界符等操作都可以在配置文件中设置。

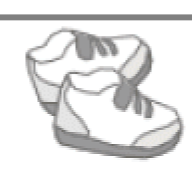
通常将配置文件定义到一个单独的文件中，在需要使用时直接通过包含语句调用即可。这里将配置文件存储到 config.php 中，其代码如下：

```
<?php
define('BASE_PATH',$_SERVER['DOCUMENT_ROOT']);          //定义服务器的绝对路径
define('SMARTY_PATH','\MR\15\example\15.1\Smarty\\');   //定义Smarty目录的绝对路径
require BASE_PATH.SMARTY_PATH.'libs\Smarty.class.php';  //加载Smarty类库文件
$smarty = new Smarty;                                   //实例化一个Smarty对象
$smarty->template_dir = BASE_PATH.SMARTY_PATH.'templates/'; //定义模板文件存储位置
$smarty->compile_dir = BASE_PATH.SMARTY_PATH.'templates_c/';//定义编译文件存储位置
$smarty->config_dir = BASE_PATH.SMARTY_PATH.'configs/';     //定义配置文件存储位置
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';        //定义缓存文件存储位置
/*  定义定界符  */
$smarty->left_delimiter = '<{';
$smarty->right_delimiter = '}>';
?>
```

参数 BASE_PATH：指定服务器的绝对路径；参数 SMARTY_PATH：指定 smarty 目录的绝对路径；参数 require：加载 Smarty 类库文件 Smarty.class.php；参数$smarty：实例化 Smarty 对象；参数$smarty->template_dir：定义模板目录存储位置；参数$smarty-> compile_dir：定义编译目录存储位置；参数$smarty-> config_dir：定义配置文件存储位置；参数$smarty-> cache_dir：定义模板缓存目录；参数$smarty->left_delimiter：定义 Smarty 使用的开始定界符；参数$smarty-> right_delimiter：定义 Smarty 使用的结束定界符。

指点迷津：

有关定界符的使用，开发者可以指定任意的格式，也可以不指定。使用 Smarty 默认的定界符“{”和“}”。



到此，Smarty 的配置讲解完毕。至于将配置文件存储在什么位置，可以根据实际情况而定。

15.1.4 Smarty 模板的应用

Note

了解了 Smarty 模板引擎的下载、安装和配置之后，下面应用 Smarty 模板开发一个实例，通过这个实例体现不出 Smarty 模板开发 Web 程序的优势，但是能让读者了解应用 Smarty 模板开发 Web 程序的流程。

例 15.1 通过 Smarty 模板开发一个实例。（**实例位置：配套资源\mr\15\example\15.1**）

其具体操作步骤如下：

（1）安装 Smarty 模板引擎。将下载的 libs 文件夹存储于服务器根目录下的 15\example\15.1\Smarty 文件夹中。

（2）在 Smarty 目录下新建 4 个目录：templates、templates_c、configs 和 cache。分别用于存储模板文件、编译文件、配置文件和缓存文件。

（3）配置 Smarty 模板。在根目录 15\example\15.1 文件夹下创建 config.php 文件，对 Smarty 模板进行配置。关于配置文件的具体内容可参考 15.1.3 节的相关内容。

（4）在 15.1 文件夹下创建 index.php 文件。首先通过 include 语句包含 config.php 文件，对 Smarty 模板引擎进行配置，然后通过 Smarty 中的 assign 方法向模板中传递数据，最后通过 display 方法指定模板页。代码如下：

```
<?php
include("config.php");
/*   使用Smarty赋值方法将一对名称/方法发送到模板中   */
$smarty->assign('title','走进Smarty模板引擎');
/*   显示模板   */
$smarty->display('index.html');
?>
```

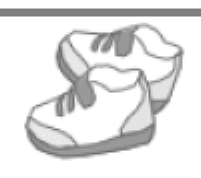
（5）在\15.1\Smarty\templates 文件夹下新建一个 index.html 静态页，获取 Smarty 模板变量传递的数据，代码如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title><{$title}></title>
</head>
<body>
<img src="images/bg.jpg" width="698" height="237" />
</body>
</html>
```

指点迷津：

在 index.html 静态页中，大括号“<{}>”为 Smarty 标签的定界符；$title 为从 PHP 动态页中传递的变量。

至此已经走进 Smarty 模板，先来看一下这个实例的文件夹架构，如图 15.5 所示。



第一个 Smarty 程序的运行结果如图 15.6 所示。

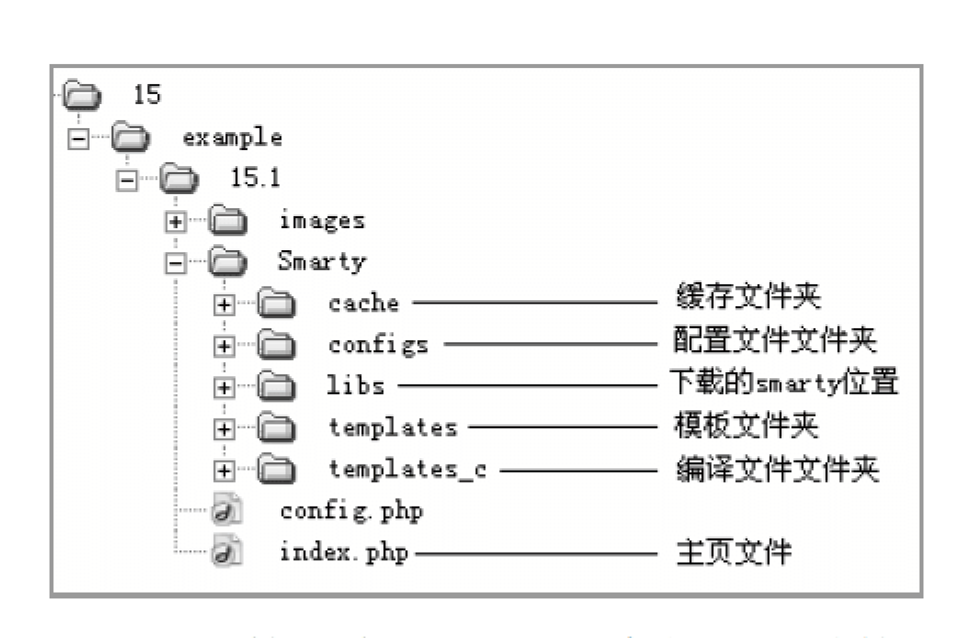


图 15.5　第一个 Smarty 程序的目录结构



图 15.6　第一个 Smarty 程序的运行结果



上机演练

上机演练 1　封装 Smarty 模板的配置方法

15.1.3 节中讲解的 Smarty 模板配置方法有一个弊端，即如果存储实例的文件夹发生变化，那么就必须修改 config.php 配置文件中常量 SMARTY_PATH 的值，否则将找不到 Smarty 类库的存储位置。这里向读者介绍另外一种 Smarty 模板的配置方法，使用这种方法后无论实例的文件夹如何改变，都不会影响到实例的运行。

15.2　Smarty 模板设计——静态页处理

视频讲解：配套资源\mr\15\video\ Smarty 模板设计——静态页处理.exe

既然 Smarty 可以将用户界面和 PHP 代码实现分离，那么应用 Smarty 模板开发程序也同样包含两部分内容：Smarty 模板设计和 Smarty 程序设计。

Smarty 模板设计的所有操作都是在模板文件中进行的，那么什么是 Smarty 模板文件呢？Smarty 模板文件是由一个页面中所有的静态元素，加上一些定界符“{…}”（Smarty 默认的定界符，读者也可以自行设置定界符）组成的，其统一存储于 templates 目录下（读者可以任意修改这个存储位置，只要路径配置正确即可）。需要注意的是，模板文件中不允许出现 PHP 代码段。Smarty 模板中的所有注释、变量、函数等都要包含在定界符内。

15.2.1　基本语法（注释、函数和属性）

在 Smarty 的基本语法中包括 3 项内容：注释、函数和属性。

1．注释

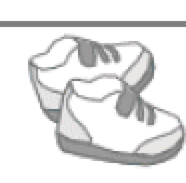
Smarty 注释包含在两个星号“*”中间，其格式如下：

```
{* 这是注释 *}
```

Smarty 注释不会在模板文件的最后输出中出现，它只是模板内在的注释。

2．函数

每一个 Smarty 标签输出一个变量或者调用某一个函数。在定界符内函数及其属性将被处



理和输出。例如：

```
{funcname attr1="val" attr2="val"}.
```

在模板文件中无论是内建函数还是自定义函数都有相同的语法。

内建函数在 Smarty 内部工作，例如{if}、{section}和{foreach}等，它们只可以使用，不能被修改。





自定义函数通过插件机制起作用，它们是附加函数，读者可以自行编辑、修改和添加，例如{assign}和{counter}。

3. 属性

大多数函数都有自己的属性以便于明确说明或修改它们的行为。Smarty 函数的属性类似于 HTML 中的属性。

在定义属性值时，如果是静态数值则不需要加引号；如果是字符串则建议使用引号；如果属性值是变量则无须加引号。

15.2.2 Smarty 模板设计变量

Smarty 有几种不同类型的变量，变量的类型取决于它的前缀是什么符号（或者被什么符号包围）。

Smarty 的变量可以直接被输出或者作为函数属性和修饰符（modifiers）的参数，或者用于内部的条件表达式等。

如果要输出一个变量，只需用定界符将它括起来即可。例如：

```
{$title}
{$array[row].id}
<body bgcolor="{#bgcolor#}">
```

1. 来自 PHP 页面中的变量

获取 PHP 页面中的变量与在 PHP 中是相同的，也需要使用“$”符号，略有不同的是对数组的读取。在 Smarty 中读取数组有两种方法：一种是通过索引获取，和 PHP 相似，可以是一维，也可以是多维；另一种是通过键值获取数组元素，这种方法的格式和以前接触过的不太一样，是使用符号“.”作为连接符。

例如，有一数组$arr=array{'user'=>'明日科技','pass'=>'mrsoft','address'=>'长春市'}，如果要获取 user 的值，表达式的格式应为$arr.user。该格式同样适用于二维数组。

调用模板内 assign 函数分配的变量也是如此，使用“$”符合加变量名来调用。

2. 从配置文件读取变量

Smarty 模板中，在配置文件中也可以定义变量。调用配置文件中变量的格式有以下两种：

☑ 使用“#”号，将变量名置于两个“#”号中间，即可像普通变量一样调用配置文件内容。

☑ 使用保留变量中的$smarty_config 来调用配置文件。

3. 保留变量

在 Smarty 模板中使用保留变量时，无须使用 assign 方法传值，直接调用变量名即可。Smarty 中常用的保留变量如表 15.1 所示。

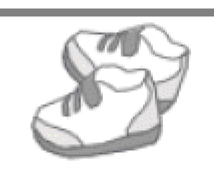




表 15.1　Smarty 中常用的保留变量

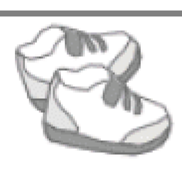
保 留 变 量	说　　明
get、post、server、session、cookie、ruquest	等价于 PHP 中的$_GET、$_POST、$_SEVER、$_COOKIE、$_REQUEST
now	当前的时间戳。等价于 PHP 中的 time()
const	用 const 包含修饰的为常量
config	配置文件内容变量

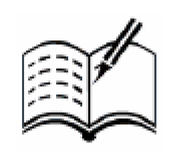
15.2.3　变量调节器

变量调节器作用于变量、自定义函数和字符串。使用“|”符号和调节器名称调用调节器。变量调节器由赋予的参数值决定其行为。参数由“:”符号分开。Smarty 中提供的变量调节器如表 15.2 所示。

表 15.2　Smarty 中提供的变量调节器

名　　称	作　　用
capitalize	将变量中的所有单词首字母大写
count_characters	计算变量中的字符数。参数值 True，表示计算空格字符，默认为 False
cat	将 cat 中的值连接到给定的变量后面
count_paragraphs	计算变量中的段落数量
count_sentences	计算变量中句子的数量
count_words	计算变量中的词数
date_format	格式化从函数 strftime()获得的时间和日期。UNIX 或 MySQL 等的时间戳（parsable by strtotime）都可以传递到 smarty。设计者可以使用 date_format 完全控制日期格式 参数 1 是输出日期的格式；参数 2 是输入为空时默认的时间格式
default	为空变量设置一个默认值。当变量为空或未分配时，由给定的默认值替代输出
escape	用于 html 转码，url 转码，在没有转码的变量上转换单引号，十六进制转码，十六进制美化，或者 javascript 转码。默认是 html 转码 参数 1 指定使用何种编码格式
indent	在每行缩进字符串，默认是 4 个字符。参数 1 作为可选参数，可以指定缩进字符数；参数 2 同样为可选参数，可以指定缩进用什么字符代替。注意：如果在 HTML 中使用缩进，那么需要使用 (空格)来代替缩进，否则没有效果
lower	将变量字符串小写
nl2br	所有的换行符将被替换成 ，功能与 PHP 中的 nl2br()函数相同
regex_replace	寻找和替换正则表达式。参数 1 指定用来替换的文本字串
replace	搜索和替换字符串。参数 1 是将被替换的文本字串；参数 2 是用来替换的文本字串
spacify	插空，在字符串的每个字符之间插入空格或其他的字符（串）。参数 1 指定将在两个字符之间插入的字符（串）
string_format	字符串的格式化。采用 sprintf()函数的语法。参数 1 指定使用的格式化方式
strip	用一个空格或一个给定字符替换所有重复空格、换行和制表符。注意：如果要去除模板文本中的区块，应使用 strip 函数





续表

名　　称	作　　用
strip_tags	去除“<”和“>”标签，包括“<”和“>”之间的任何内容
truncate	从字符串开始处截取指定长度的字符，默认是 80 个字节 参数 1 设置截取字符的数量；参数 2 设置截取后追加在截取词后面的字符串；参数 3 设置是截取到词的边界（False）还是精确到字符（True）
upper	将变量改为大写
wordwrap	控制段落的宽度（也就是多少个字符为一行，超过这个字符数换行），默认 80 个字节 参数 1 设置段落（句子）的宽度；参数 2 设置使用什么字符进行约束（默认是换行符\n）；参数 3 设置是约束到词的边界（False）还是精确到字符（True）

指点迷津：

如果为数组变量应用单值变量的调节，结果是数组的每个值都被调节。如果只希望调节器用一个值调节整个数组，那么必须在调节器名字前加上@符号，例如{$articleTitle|@count}（这将会在$articleTitle 数组中输出元素的数目）。

15.2.4　内建函数（动态文件、模板文件的包含和流程控制语句）

Smarty 自身定义了一些内建函数，存储于 Smarty 模板中。它是模板语言的一部分，用户不能创建名称和内建函数相同的自定义函数，也不能修改内建函数。内建函数包括：foreach、if 和 section 此类的流程控制语句，还包括像 include 和 include_php 这样的函数。本节中将讲解一些常用内建函数的使用方法，如果读者要了解全部内建函数的知识，可以参考 Smarty 手册。

1. foreach 循环控制

Smarty 模板中的 foreach 语句可以循环输出数组。与另一个循环控制语句 section 相比，在使用格式上要简单得多，一般用于简单数组的处理。foreach 语法如下：

```
{foreach name=foreach_name key=key item=item from=arr_name}
…
{/foreach}
```

参数：name 为该循环的名称；参数 key 为当前元素的键值；参数 item 是当前元素的变量名；参数 from 是该循环的数组。其中，item 和 from 是必选参数，不可省略。

2. include 函数——在模板中包含子模板

include 函数用于在当前模板中包含其他模板，当前模板中的变量在被包含的模板中可用。函数语法如下：

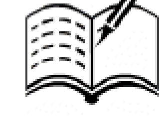
```
{include file="file_name " assign=" " var="   "}
```

参数 file 指定包含模板文件的名称，为必选参数；参数 assign 指定一个变量保存包含模板的输出；参数 var 传递给待包含模板的本地参数，只在待包含模板中有效。

3. if…elseif…else 条件语句

if 条件语句的使用和 PHP 中的 if 大同小异。需要注意的是，if 必须以/if 为结束标记。其语法格式如下：






```
{if 条件语句1}
    语句1
{elseif 条件语句2}
    语句2
{else}
    语句3
{/if}
```

在上述的条件语句中，除了可以使用 PHP 中的<、>、=、!=等常见运算符外，还可以使用 eq、ne、neq、gt、lt、lte、le、gte、ge、is even、is odd、is not even、is not odd、not、mod、div by、even by、odd by 等修饰词修饰。

4. ldelim 和 rdelim——输出大括号“{”和“}”

ldelim 和 rdelim 函数用于输出定界符，也就是大括号“{”和“}”。因为模板引擎总是尝试解释大括号内的内容，因此如果需要输出大括号，则可以使用这两个函数。

例如，在模板页面中输出一个 JavaScript 脚本，因为 JavaScript 脚本中会涉及大括号的使用，所以应用 ldelim 和 rdelim 输出 JavaScript 脚本中的大括号。代码如下：

```
<script language=javascript>
function check_form() {ldelim}
    if (user.value == ''){ldelim}
        alert('请输入用户名');
        return false;
    {rdelim}
{rdelim}
</script>
```

多学两招：

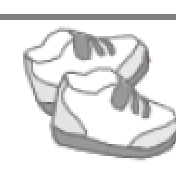
通过 literal 标签输出大括号

通过 ldelim 和 rdelim 标签可以输出 JavaScript 脚本中的大括号，该方法需要对每个大括号都进行操作。如果使用 literal 标签就没有那么麻烦了，它可以将整个标签区域内的数据当作文本处理。同样是在模板文件中输出 JavaScript 脚本，应用 literal 标签就简单多了，代码如下：

```
{literal}
<script language=javascript>
function check_form() {
    if (user.value == ''){
        alert('请输入用户名');
        return false;
    }
}
</script>
{/literal}
```

指点迷津：

如果要在 Smarty 模板文件中直接输出 JavaScript 脚本或定义 CSS 样式，并且 Smarty 使用默认的定界符“{}”，那么就会应用到上述两个函数中的一个，对 JavaScript 脚本或 CSS 样式中的大括号进行输出。



5. section 循环控制

section 是 Smarty 模板中的另一个循环语句，该语句可用于比较复杂的数组。section 的语法结构如下：



```
{section name="sec_name"loop=$arr_name start=num step=num max= show=}
```



section 语句的参数说明如表 15.3 所示。

表 15.3　section 语句的参数说明

参　数	说　明
name	循环的名称
loop	循环的数组
start	表示循环的初始位置。例如 start=2，说明循环是从 loop 数组的第二个元素开始
step	表示步长，例如 step=2，说明循环一次后数组的指针将向下移动两位，依此类推
max	设定循环的最大执行次数
show	决定是否显示该循环

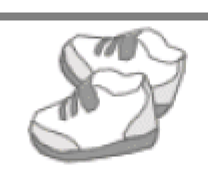
section 循环语句最擅长的是操作 ADODB 从数据库中读取到的数据，因为 ADODB 返回的数据就是一个二维数组。

15.2.5　自定义函数

Smarty 中包含很多自定义函数，通过这些自定义函数可以实现很多的功能。Smarty 中的自定义函数如表 15.4 所示。

表 15.4　Smarty 中的自定义函数

名　称	作　用
assign()	用于在模板被执行时为模板变量赋值。参数 var 被赋值的变量名；参数 value 赋给变量的值
counter()	用于输出一个记数过程。counter 保存了每次记数时的当前记数值
cycle()	用于轮转使用一组值。该特性使得在表格中交替输出颜色或轮转使用数组中的值变得很容易 参数 name 指定轮转的名称；参数 values 指定待轮转的值，可以是用逗号分隔的列表（可查看 delimiter 属性）或一个包含多值的数组；参数 print 设置是否输出值；参数 advance 设置是否使用下一个值（为 False 时使用当前值）；参数 delimiter 设置 values 属性中使用的分隔符，默认是逗号；参数 assign 指定输出值将被赋给模板变量的名称
debug()	将调试信息输出到页面上。该函数是否可用取决于 Smarty 的 debug 设置
eval()	按处理模板的方式获取变量的值。该特性可用于在配置文件的标签/变量中嵌入其他模板标签/变量
fetch()	用于从本地文件系统、HTTP 或 FTP 上取得文件并显示文件的内容。如果文件名称以“http://”开头，将取得该网站页面并显示；如果文件名称以“ftp://”开头，将从 ftp 服务器取得该文件并显示
html_checkboxes()	根据给定的数据创建复选框组





续表

名　称	作　用
html_image()	创建一个图像的 HTML 标签。如果没有提供高度和宽度值，则根据图像的实际大小自动取得
html_options()	根据给定的数据创建选项组
html_radios()	根据给定的数据创建单选按钮组
html_select_date()	创建日期下拉菜单。它可以显示任意年月日
html_select_time()	创建时间下拉菜单。它可以显示任意时分秒
html_table()	将数组中的数据填充到 HTML 表格中
math()	允许模板设计者在模板中进行数学表达式运算
mailto()	mailto 自动生成电子邮件链接，并根据选项决定是否对地址信息编码
popup()	创建 JavaScript 弹出窗口
textformat()	格式化文本。该函数主要清理空格和特殊字符

15.2.6　配置文件

配置文件的应用有利于设计者管理文件中的模板全局变量。例如，定义一个模板色彩变量。一般情况下如果想改变一个程序的外观色彩，必须更改每一个文件的颜色变量。如果有配置文件，色彩变量就可以保存在一个单独的文件中，只要改变配置文件就可以实现色彩的更新。

1. 创建配置文件

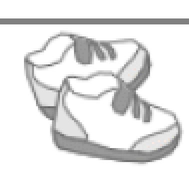
对于配置文件可以任意命名，其存储位置由 Smarty 对象的$config_dir 属性指定。如果存在不只在一个区域内使用的变量值，则可以使用三引号（"""）将它完整的封装起来。在创建配置文件时，建议在程序运行前使用“#”加一些注释信息，这样有助于程序的阅读、更新。

在配置文件中既可以声明全局变量，也可以声明局部变量。如果声明局部变量，则可以使用中括号“[]”括起来，在中括号之内声明的变量属于局部变量，而中括号之外声明的变量都是全局变量。中括号的使用不仅使配置文件中声明变量的模块变得清晰，而且可以在模板中选择加载中括号内的变量。

例如，创建一个配置文件，分别声明全局变量和局部变量。代码如下：

```
# global variables                      #在每行之前使用#，表示注释
title = "引用配置文件"                  #声明全局变量
[table]                                 #声明局部变量
border = "1"
cellpadding="1"
cellspacing="1"
bordercolor="#FFFFFF"
table_bgcolor="#333333"
[td]                                   #声明局部变量
bgcolor="#FFFFFF"
```

如果某个特定的局部变量已经载入，则全局变量和局部变量都可以载入；如果当某个变量名既是全局变量又是局部变量，则局部变量将被优先赋予值来使用；如果在一个局部中两个变量名相同，则最后一个将被赋值使用。





2. 加载配置文件

加载配置文件应用 Smarty 的内建函数 config_load，其语法如下：

```
{config_load file="file_name " section="add_attribute" scope="" global=""}
```

config_load()函数的参数说明如表 15.5 所示。

表 15.5　config_load()函数的参数说明

参　数	说　明
file	指定包含的配置文件的名称
section	附加属性，当配置文件中包含多个部分时应用，指定具体从哪一部分中取得变量
scope	加载数据的作用域，取值必须为 local、parent 或 global。local 说明该变量的作用域为当前模板；parent 说明该变量的作用域为当前模板和当前模板的父模板（调用当前模板的模板）；global 说明该变量的作用域为所有模板。当指定 scope 属性时，可以设置 global 属性，但模板忽略该属性值，而以 scope 属性为准
global	说明加载的变量是否全局可见，等同于 scope=parent

3. 引用配置文件中的变量

当配置文件加载成功后，就可以在模板中引用配置文件中声明的变量了。引用配置文件应用的是“#”或 Smarty 的保留变量$smarty.config。其应用示例如下：

```
{ config_load file="file_con.conf"}          {*  加载配置文件  *}
{#title#}
    <td height="228" colspan="2" align="left" valign="top" class="{$smarty.config.styles}">
```

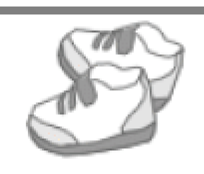
上机演练

上机演练 2　Smarty 模板中日期、时间的格式化输出

在 PHP 脚本中日期、时间的格式化输出最常用的就是 date()函数，那么在 Smarty 模板中该如何完成日期、时间的输出呢？这就是我们要练习的内容，通过 Smarty 模板中的 date_format 函数完成日期、时间的格式化输出，其运行结果如图 15.7 所示。



图 15.7　Smarty 模板中日期、时间的格式化输出



Note

上机演练 3　Smarty 模板中的编码

在 PHP 脚本中应用 urlencode()函数对超链接中传递的参数进行编码，而在 Smarty 模板中提供 escape 函数对字符串进行编码。通过它完成对超链接中传递的参数进行编码，其运行结果如图 15.8 所示。



图 15.8　Smarty 模板中对超链接的参数值进行编码

上机演练 4　Smarty 模板制作日期、时间选择器

在 Smarty 模板中，提供自定义的日期、时间处理函数，通过它们可以获取到不同格式的日期、时间信息。应用 Smarty 模板中的日期时间选择函数，设计一个日期时间选择器，记录用户登录的时间，其运行结果如图 15.9 所示。

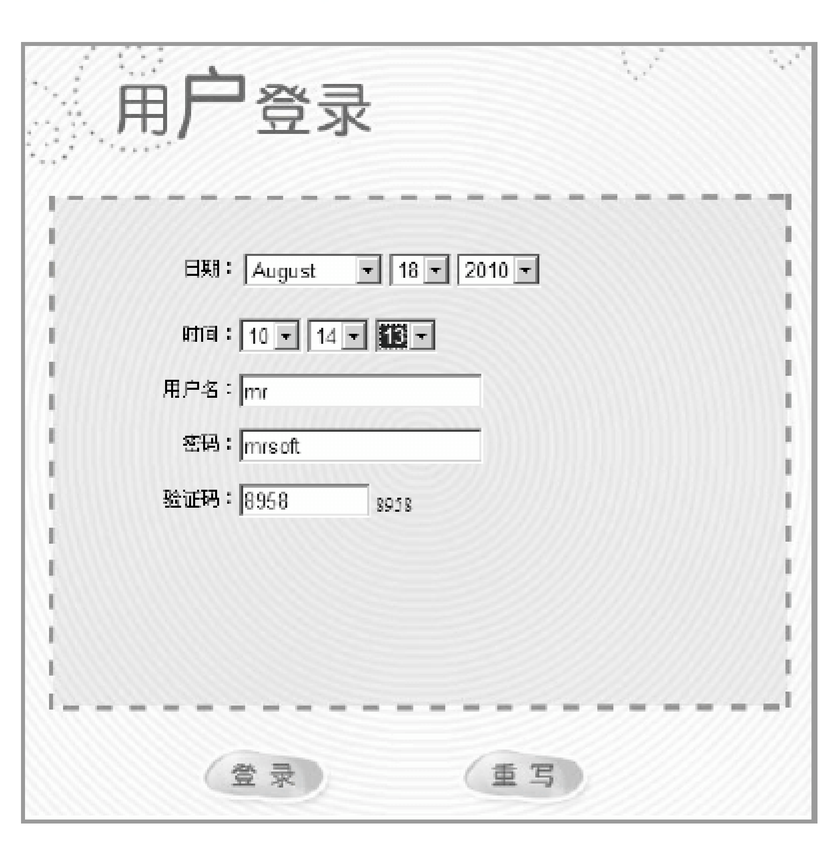


图 15.9　Smarty 模板中的日期、时间选择器

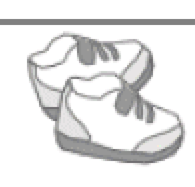
15.3　Smarty 程序设计——动态文件操作

视频讲解：配套资源\mr\15\video\ Smarty 程序设计——动态文件操作.exe

Smarty 程序设计在动态 PHP 文件中进行操作，其功能可以分为两种：一种功能是配置 Smarty，如变量 template_dir、$config_dir 等；另一种功能是和 Smarty 模板之间的交互，如方法 assign、display 等。

15.3.1　SMARTY_PATH 常量

SMARTY_PATH 常量定位 Smarty 类文件的完整系统路径，如果没有定义 Smarty 目录，则



Smarty 将会试着自动创建合适的值。如果定义了路径，则必须要以反斜杠结束。该常量的应用是在 Smarty 的配置文件中，通过它获取 Smarty 类的绝对路径。

例如，在 15.1.3 节创建的配置 config.php 中就应用到了这个常量。其关键代码如下：





```
define('BASE_PATH',$_SERVER['DOCUMENT_ROOT']);            //定义服务器的绝对路径
define('SMARTY_PATH','\MR\15\example\15.1\Smarty\\');      //定义Smarty目录的绝对路径
require BASE_PATH.SMARTY_PATH.'libs\Smarty.class.php';     //加载Smarty类库文件
```

15.3.2 Smarty 程序设计变量

在 Smarty 中提供了很多的变量，这里只讲解比较常用的几个，如果读者想详细了解 Smarty 变量，可参考 Smarty 的手册。

☑ $template_dir：模板目录。模板目录用来存放 Smarty 模板，在前面的实例中，所有的.html 文件都是 Smarty 模板。模板的后缀没有要求，一般为.html、.tpl 等。

☑ $compile_dir：编译目录。顾名思义，就是编译后的模板和 PHP 程序所生成的文件默认路径为当前执行文件所在的目录下的 templates_c 目录。进入到编译目录，可以发现许多“%%…%%index.html.php”格式的文件。随便打开一个这样的文件可以发现，实际上 Smarty 将模板和 PHP 程序又重新组合成一个混编页面。

☑ $cache_dir：缓存目录。用来存放缓存文件。同样，在 cache 目录下可以看到生成的.html 文件。如果 caching 变量开启，那么 Smarty 将直接从这里读取文件。

☑ $config_dir：配置目录。该目录用来存放配置文件。

☑ $debugging：调试变量。该变量可以打开调试控制台。只要在配置文件（config.php）中将$smarty->debugging 设为 True 即可使用。

☑ $caching：缓存变量。该变量可以开启缓存。只要当前模板文件和配置文件未被改动，Smarty 就直接从缓存目录中读取缓存文件而不重新编译模板。

15.3.3 Smarty 方法

在 Smarty 提供的方法中最常用的为 assign 方法和 display 方法。

1. assign 方法

assign 用于在模板被执行时为模板变量赋值。其语法如下：

```
{assign var=" " value=" "}
```

参数 var 是被赋值的变量名；参数 value 是赋给变量的值。

2. display 方法

display 方法用于显示模板，需要指定一个合法的模板资源的类型和路径。另外，还可以通过第二个可选参数指定一个缓存号，相关的信息可以查看缓存。

```
void display (string template [, string cache_id [, string compile_id]])
```

参数 template 指定一个合法的模板资源的类型和路径。参数 cache_id 为可选参数，指定一个缓存号。参数 compile_id 为可选参数，指定编译号。编译号可以将一个模板编译成不同版本使用，例如针对不同的语言编译模板。编译号的另外一个作用是，如果存在多个$template_dir



模板目录，但只有一个$compile_dir 编译后存档目录，这时可以为每一个$template_dir 模板目录指定一个编译号，以避免相同的模板文件在编译后互相覆盖。相对于在每一次调用 display()时都指定编译号，也可以通过设置$compile_id 编译号属性来一次性设定。

15.3.4　Smarty 缓存

在讲解 Smarty 的缓存之前，先将它和 Smarty 的编译过程进行一个对比，让读者明白缓存到底意味着什么。

Smarty 的编译功能默认是开启的，而 Smarty 缓存则必须由开发人员来开启。

编译的过程是将模板转换为 PHP 脚本，虽然在模板没有被修改的情况下，不会重新执行转换过程，但这个编译过的模板其实就是一个 PHP 脚本，只是减少了模板转换的压力，仍需要在逻辑层执行获取数据的操作，而这个获取数据的操作是耗费内存最大的。

缓存不仅将模板转换为 PHP 脚本，而且还将模板内容转换为静态页面，这不仅减少了模板转换的压力，而且不再需要在逻辑层执行获取数据的操作。

这就是 Smarty 的缓存机制，它是一种更加理想的开发 Web 程序的方法。下面就来学习这种方法。

1. 创建缓存

开启缓存的方法非常简单，将 Smarty 对象中$config 的值设置为 True，同时通过 Smarty 对象中的$cache_dir 属性指定缓存文件的存储位置即可。其操作代码如下：

```
$smarty->caching=true;                                          //开启缓存
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';            //定义缓存文件存储位置
```

2. 缓存的生命周期

缓存创建成功后，必须为它设置一个生命周期，如果它一直不更新，那么就没有任何意义。设置缓存生命周期应用的是 Smarty 对象中的$cache_lifetime 属性，缓存时间以秒为单位，默认值是 3600 秒。其操作代码如下：

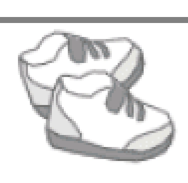
```
$smarty->caching=true;                                          //开启缓存
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';            //定义缓存文件存储位置
$smarty->cache_lifetime=3600                                    //设置缓存时间为1小时
```

如果将$caching 的值设置为 2，那么就可以控制单个缓存文件各自的过期时间。

3. 同一模板生成多个缓存

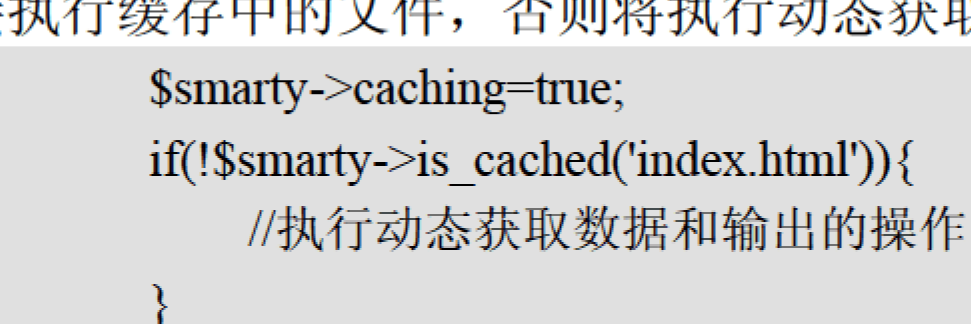
在实际的程序开发中，经常会遇到这样的情况，即：同一个模板文件生成多个页面。而此时要对这多个页面进行缓存，应用的是 Smarty 中的 display 方法，通过该方法的第二个参数设置缓存号，有几个不同的缓存号就有几个缓存页面。操作代码如下：

```
$smarty->caching=true;                                          //开启缓存
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';            //定义缓存文件存储位置
$smarty->cache_lifetime=3600;                                   //设置缓存时间为1小时
$smarty->display('index.html',$_GET['id']);                     //将id作为第二个参数传递
```



4. 判断模板文件是否已被缓存

如果页面已经被缓存，那么就可以直接调用缓存文件，而不再执行动态获取数据和输出的操作。为了避免在开启缓存后再次执行动态获取数据和输出操作给服务器带来的压力，最佳的方法就是应用 Smarty 对象中的 is_cached 方法，判断指定的模板是否存在缓存，如果存在则直接执行缓存中的文件，否则将执行动态获取数据和输出的操作。操作代码如下：



```
$smarty->caching=true;                                    //开启缓存
if(!$smarty->is_cached('index.html')){
    //执行动态获取数据和输出的操作
}
$smarty->display('index.html');
```

多学两招：

如何判断同一模板中的多个缓存是否存在?

判断同一模板中的多个缓存是否存在与判断同一模板生成多个缓存类似，都是以缓存号为依据。判断同一模板的多个缓存是否存在应用 is_cached 方法，通过该方法的第二个参数设置缓存号，即可判断出对应的缓存是否存在。其方法如下：

```
$smarty->caching=true;                                    //开启缓存
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';      //定义缓存文件存储位置
$smarty->cache_lifetime=3600;                             //设置缓存时间为 1 小时
if(!$smarty->is_cached('index.html',$_GET['id'])){
    //执行动态获取数据和输出的操作
}
$smarty->display('index.html',$_GET['id']);               //将 id 作为第二个参数传递
```

5. 清除模板中缓存

缓存的清除有两种方法：

（1）第一种是 clear_all_cache 方法，清除所有模板缓存。其语法如下：

```
void clear_all_cache (int expire time)
```

可选参数 expire time，可以指定一个以秒为单位的最小时间，超过这个时间的缓存都将被清除。

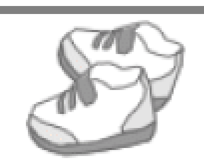
（2）第二种是 clear_cache 方法，清除指定模板的缓存。

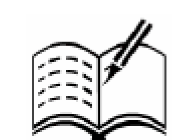
```
void clear_cache (string template [, string cache id [, string compile id [, int expire time]]])
```

如果这个模板有多个缓存，可以使用第二个参数指定要清除缓存的缓存号，同时还可以通过第三个参数指定编译号。可以把模板分组，以便可以方便地清除一组缓存。第四个参数是可选的，用来指定超过某一时间（以秒为单位）的缓存才会被清除。

例如，分别应用这两种方法清除缓存。代码如下：

```
$smarty->caching=true;                          //开启缓存
$smarty->clear_all_cache();                     //清除所有缓存
$smarty->clear_cache('index.html');             //清除index.html模板的缓存
$smarty->clear_cache('index.html','$_GET['id']');  //清除index.html模板中一个指定缓存号的缓存
$smarty->display('index.html');
```





上机演练

上机演练 5　通过 section 循环输出数据

在 PHP 动态页中，可以通过 while、do…while、for 和 foreach 语句实现数据的循环输出，而在 Smarty 中它也有属于自己的循环输出语句 foreach 和 section。本例通过 section 语句完成数据的循环输出，其运行结果如图 15.10 所示。



图 15.10　Smarty 模板下载页面

上机演练 6　开启网站注册页面的缓存

缓存的合理应用应以实际的开发需要为依据，对于那些需要经常更新的程序是否开启缓存，以及如果开启缓存，周期长短设置多长时间等，都必须根据程序的实际需求进行设置。在注册页面应用缓存技术，注册页面的运行结果如图 15.11 所示。



图 15.11　注册页面的运行结果

上机演练 7　通过配置文件定义变量

配置文件的应用有利于设计者管理文件中的模板全局变量，例如定义一个模板色彩变量。一般情况下，如果想改变一个程序的外观色彩，必须更改每一个文件的颜色变量。如果有配置文件，色彩变量就可以保存在一个单独的文件中，只要改变配置文件就可以实现色彩的更新，这与在 Web 页面开发中应用的 CSS 样式非常相似。应用 Smarty 中的配置文件，并通过配置文件定义页面中 body 标签的样式，其运行结果如图 15.12 所示。






图 15.12 通过配置文件定义页面的样式

本 章 摘 要

1．介绍 Smarty 模板的安装、配置及使用。
2．Smarty 的模板设计和程序设计。
3．Smarty 模板的应用。

习　　题

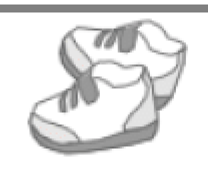
1．对 Smarty 模板描述错误的是（　　）。
　A．Smarty 模板是 zend 完全官方化的模板引擎
　B．Smarty 模板具有强大的表现逻辑功能
　C．Smarty 模板的特点是只适用于 PHP 程序员
　D．Smarty 模板具有模板编译、缓存等优秀的特征

2．在安装 Smarty 模板时，创建存储 Smarty 模板的目录是（　　）。
　A．templates　　B．templates_c　　C．configs　　D．cache

3．Smarty 模板引擎需要在 PHP 的应用程序逻辑和页面模板中配合使用，才能完全分离表现层和逻辑层。在 PHP 程序中以下哪个步骤执行了编译的过程，将模板转换成 PHP 脚本？（　　）
　A．加载 Smarty 模板引擎类，建立 Smarty 对象
　B．修改 Smarty 的默认行为
　C．将程序中动态获取的变量，通过 Smarty 对象中的 assign 方法置入模板中
　D．利用 Smarty 对象中的 display 方法将模板内容输出

4．以下在 Smarty 中使用变量的方式哪个是不正确的？（　　）
　A．{func var="test $foo test"}　　B．{func var="test $foo [0] test"}
　C．{func var="test $foo[bar] test"}　　D．{func var="test $foo.bar test"}

5．对 smarty 模板引擎来说，缓存是必不可少的，下面哪一个是和缓存控制无关的 smarty 属性？（　　）
　A．$smarty->caching　　B．$smarty->cache_dir



C．$smarty->cache_lifetime　　　　　　D．$smarty->is_cache

6．模板文件需要存储在（　　）文件夹。

7．开启缓存的代码为（　　）。

8．显示模板文件需要使用（　　）函数。

9．为模板变量赋值需要使用（　　）函数。

10．以下代码是模板页的相关内容，在 PHP 文件中为此循环赋值的代码为（　　）。

```
{foreach from=$value item=key}
        <tr>
            <td>{$key[0] }</td><td>{ $key[1] }</td><td>{ $key[2] }</td><td>{ $key[3] }</td>
        </tr>
{/foreach}
```

Note

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 14 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　Smarty+ADODB 完成数据的分页显示

通过面向对象的方法完成 Smarty 的配置，ADODB 连接 MySQL、操作 MySQL 数据库以及分页的功能，其运行结果如图 15.13 所示。



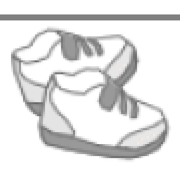
图 15.13　Smarty+ADODB 完成数据的分页显示

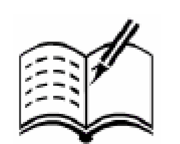
实战模拟 2　Smarty 模板中 truncate 方法截取字符串

在 Smarty 中，提供 truncate 方法对字符串进行截取，该方法会截取到一个词的末尾。应用 truncate 方法对论坛中标题和内容进行截取，并且应用省略号替换截取的内容，其运行结果如图 15.14 所示。

实战模拟 3　Register_function 方法注册模板函数

在 Smarty 中，注册模板函数可以实现与 PHP 中自定义函数相同的功能，其特点是模板函





数的注册在动态页中完成，实际的应用在静态页进行中，完全体现 Smarty 的动静分离的开发模式。分页输出论坛帖子的信息，并通过注册模板函数对帖子的内容进行截取，如果帖子内容超出指定的长度就对其进行截取，并且使用省略号替代被截取部分，其运行结果如图 15.15 所示。



图 15.14　truncate 方法截取字符串



图 15.15　注册模板函数截取字符串

实战模拟 4　Smarty 模板中的关键字描红技术

在 Smarty 模板中同样可以实现关键字描红的功能，其应用的是 Smarty 中的 replace 变量。在订单查询模块，实现对查询关键字的描红功能，其运行结果如图 15.16 所示。

实战模拟 5　Smarty 模板中生成数字验证码

应用 Smarty 模板中的 foreach 语句在模板页中直接生成一个数字验证码，它比在动态 PHP 中开发的验证码更加简单、实用。Smarty 模板中生成的验证码运行结果如图 15.17 所示。



订单管理=>订单查询

订单号：33007　提交　重置

ID	订单号	收货人	时间	操作
1	010603300761	明日科技	2010-08-12 19:50:39	未处理
2	010603300762	明日科技	2010-08-12 19:51:22	已处理
3	010603300763	明日科技	2010-08-12 19:50:39	未处理
4	010603300764	明日科技	2010-08-12 19:51:22	已处理
5	010603300765	明日科技	2010-08-12 19:50:39	未处理
6	010603300766	明日科技	2010-08-12 19:51:22	已处理
7	010603300767	明日科技	2010-08-12 19:50:39	未处理
8	010603300768	明日科技	2010-08-12 19:51:22	已处理

图 15.16　Smarty 模板中的关键字描红



图 15.17　Smarty 模板中生成数字验证码



ThinkPHP 框架

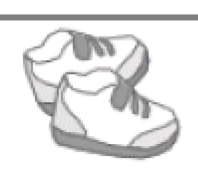
（自学视频、源程序：配套资源\mr\16\）

ThinkPHP 是一个免费开源的，快速、简单的，面向对象的轻量级 PHP 开发框架，遵循 Apache2 开源协议发布，是为了敏捷 Web 应用开发和简化企业级应用开发而诞生的。

ThinkPHP 借鉴国外很多优秀的框架和模式，使用面向对象的开发结构和 MVC 模式，采用单一入口模式等，融合了 Struts 的 Action 思想、JSP 的 TagLib（标签库）和 RoR 的 ORM 映射及 ActiveRecord 模式，封装了 CURD 和一些常用操作，在项目配置、类库导入、模板引擎、查询语言、自动验证、视图模型、项目编译、缓存机制、SEO 支持、分布式数据库、多数据库连接和切换、认证机制和扩展性方面均有独特的表现。通过本章的学习，读者将对 ThinkPHP 框架有深入的认识，并且能够达到可以简单应用的程度。

学习摘要：

- ThinkPHP 概述
- ThinkPHP 项目创建流程
- ThinkPHP 项目目录结构、部署方案、命名规范和构建流程
- ThinkPHP 的配置
- ThinkPHP 的控制器
- ThinkPHP 的模型，模型的命名、实例化、属性访问
- ThinkPHP 的模型，连接数据库、创建数据、连贯操作以及 CURD 操作
- ThinkPHP 的视图
- ThinkPHP 的内置模板引擎



16.1 ThinkPHP 简介



ThinkPHP 可以更方便和快捷地开发和部署应用。其不仅仅是企业级应用，任何 PHP 应用开发都可以从 ThinkPHP 的简单和快速的特性中受益。ThinkPHP 本身具有很多的原创特性，并且倡导“大道至简、开发由我”的开发理念，用最少的代码完成实现的功能。

ThinkPHP 遵循 Apache2 开源许可协议发布，意味着用户可以免费使用 ThinkPHP，甚至允许把基于 ThinkPHP 开发的应用开源或商业产品发布/销售。

16.1.1 ThinkPHP 框架的特点

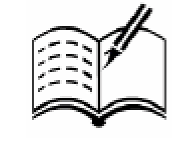
ThinkPHP 是一个性能卓越并且功能丰富的轻量级 PHP 开发框架，其宗旨就是让 Web 应用开发更简单、快速。ThinkPHP 值得推荐的特性包括以下几点。

- ☑ 类库导入：ThinkPHP 首先采用基于类库包和命名空间的方式导入类库，让类库导入看起来更加简单清晰，而且还支持冲突检测和别名导入。为了方便项目的跨平台移植，系统还可以严格检查加载文件的大小写。
- ☑ URL 模式：系统支持普通模式、PATHINFO 模式、REWRITE 模式和兼容模式的 URL 方式，支持不同的服务器和运行模式的部署，配合 URL 路由功能，可以随心所欲地构建需要的 URL 地址和进行 SEO 优化工作。
- ☑ 编译机制：独创的核心编译和项目的动态编译机制，有效地减少了 OOP 开发中文件加载的性能开销。
- ☑ 查询语言：内建丰富的查询机制，包括组合查询、复合查询、区间查询、统计查询、定位查询、动态查询和原生查询，让数据查询简捷高效。
- ☑ 视图模型：轻松、动态地创建数据库视图，使得多表查询易实现。
- ☑ 分组模块：不用担心大项目的分工协调和部署问题，同时可以解决跨项目的难题。
- ☑ 模板引擎：系统内建了一款卓越的基于 XML 的编译型模板引擎，支持两种类型的模板标签，融合了 Smarty 和 JSP 标签库的思想，支持标签库扩展。另外，通过驱动还可以支持 Smarty、EaseTemplate、TemplateLite、Smart 等第三方模板引擎。
- ☑ AJAX 支持：内置 AJAX 数据返回方法，支持 JSON、XML 和 EVAL 格式返回客户端，并且系统不绑定任何 AJAX 类库，可随意使用自己熟悉的 AJAX 类库进行操作。
- ☑ 缓存机制：系统支持包括文件方式、APC、Db、Memcache、Shmop、eAccelerator 和 XCache 在内的多种动态数据缓存类型以及可定制的静态缓存规则，并提供了快捷方法进行存取操作。

16.1.2 环境要求

ThinkPHP 可以支持 Windows/UNIX 服务器环境，可运行于包括 Apache、IIS 在内的多种 Web 服务器上。需要 PHP5.0 及以上版本支持，同时支持 MySQL、MS SQL、PgSQL、SQLite、Oracle 等数据库。





16.1.3　下载 ThinkPHP 框架

获取 ThinkPHP 的方式有很多种。其官方网址为：http://thinkphp.cn。SVN 的下载地址为：完整版本 http://thinkphp.googlecode. com/svn/trunk，核心版本 http://thinkphp.googlecode.com/svn/trunk/ThinkPHP。

☎小测试

1. 什么是 MVC

MVC 是一种经典的程序设计理念，此模式将应用程序分为 3 个部分：模型层（Model）、视图层（View）和控制层（Controller），MVC 是这 3 个部分英文字母的缩写。

指点迷津：

MVC 设计模式产生的原因：应用程序中用来完成任务的代码——模型层（也称业务逻辑），通常是程序中相对稳定的部分，重用率高；而与用户的交互界面——视图层，却经常改变。如果因需求变动而不得不对业务逻辑代码修改，或者要在不同的模块中应用到相同的功能而重复编写业务逻辑代码，不仅会降低整体程序开发的进度，也会使未来的维护变得非常困难。因此，将业务逻辑代码与外观分离，将会更方便地根据需求改进程序，这就是 MVC 设计模式。

在 PHP Web 开发中，MVC 设计模式的各自功能及相互关系如图 16.1 所示。

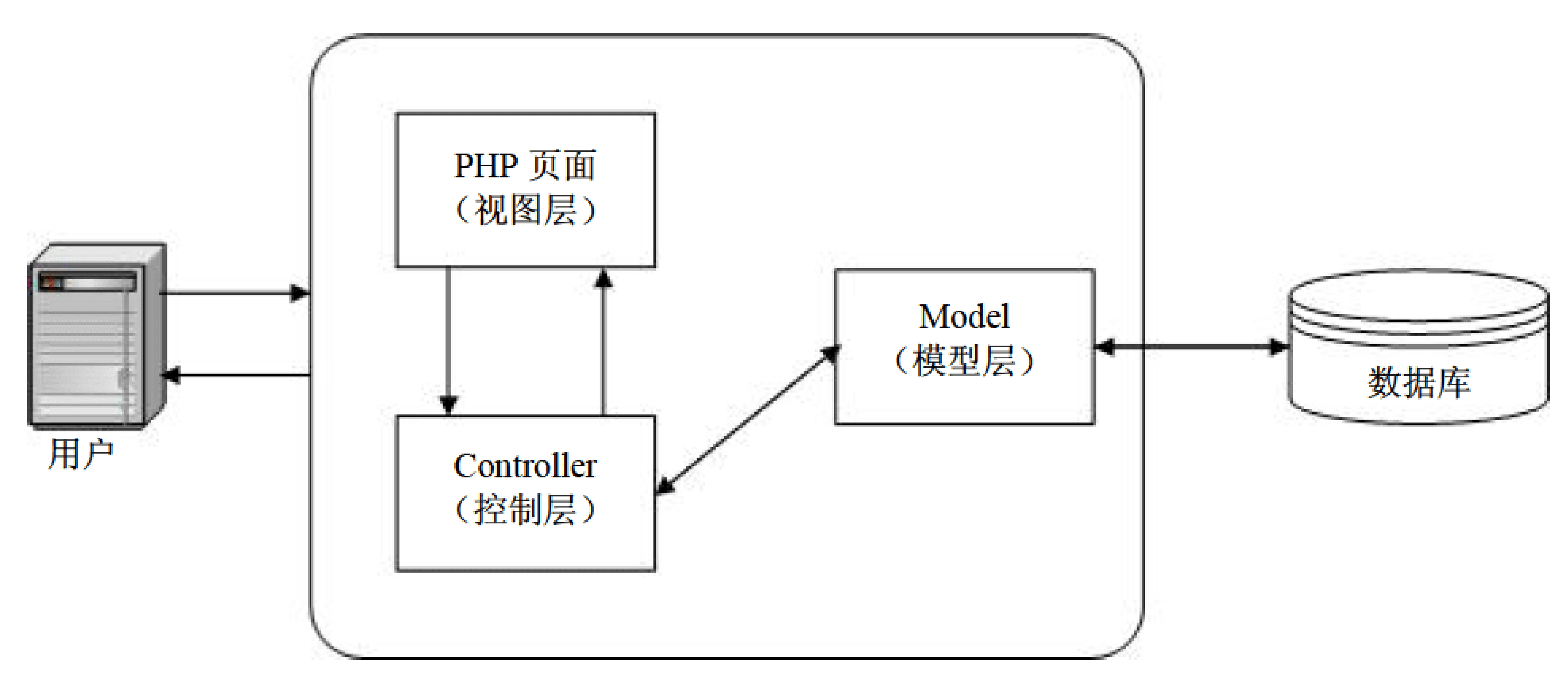


图 16.1　MVC 关系图

☑　模型层（Model）

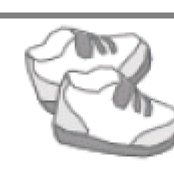
模型层是应用程序的核心部分，它可以是一个实体对象或一种业务逻辑。它之所以称为模型，是因为它在应用程序中有更好的重用性和扩展性。

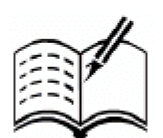
☑　视图层（View）

视图层提供应用程序与用户之间的交互界面，在 MVC 理论之中，这一层并不包含任何的业务逻辑，仅提供一种与用户交互的视图。

☑　控制层（Controller）

控制层用于对程序中的请求进行控制，作用就如同宏观调控，它可以选择调用哪些视图或者调用哪些模型。







2. 什么是 CURD

CURD 是数据库操作的缩写，也是几种数据库操作技术的缩写，其中，“C”代表创建（Create），“U”代表更新（Update），R 代表读取（Read），D 代表删除（Delete）。CURD 定义了用于处理数据的基本操作。之所以将 CURD 提升到一个技术难题的高度，是因为完成一个涉及在多个数据库系统中进行 CURD 操作的汇总相关的活动，其性能可能会随数据关系的变化而有非常大的差异。

CURD 在具体的应用中并非一定使用 create、update、read 和 delete 字样的方法，但是它们完成的功能是一致的。例如，ThinkPHP 就是使用 add、save、select 和 delete 方法表示模型的 CURD 操作。

3. 什么是单一入口

单一入口通常是指一个项目或应用具有一个统一（但并不一定是唯一）的入口文件，也就是说，项目的所有功能操作都是通过该入口文件进行的，并且往往入口文件是第一步被执行的。

单一入口的优点是项目整体比较规范，因为同一个入口，往往其不同操作之间具有相同的规则。另外一个方面就是单一入口控制较为灵活，因为拦截方便，如权限控制、用户登录方面的判断和操作可以统一处理。

16.2 ThinkPHP 架构

视频讲解：配套资源\mr\16\video\ThinkPHP 架构.exe

ThinkPHP 遵循简洁实用的设计原则，在兼顾开发速度和执行速度的同时，也注重易用性。本节将对 ThinkPHP 框架的整体思想和架构体系进行详细说明。

16.2.1 ThinkPHP 的目录结构

在 ThinkPHP 框架中，目录分为两部分：系统目录和项目目录。系统目录是下载的 ThinkPHP 框架类库本身的，如表 16.1 所示。

表 16.1 系统目录

目 录 名 称	主 要 作 用
Common	包含框架的一些公共文件、系统定义和惯例配置等
Lang	目录语言文件夹，目前 ThinkPHP 支持的语言有简体中文、繁体中文和英文
Lib	系统的基类库目录
Tpl	系统的模板目录
Mode	框架模式扩展目录
Vendor	第三方类库目录

项目目录是用户实际应用的目录，如表 16.2 所示（ThinkPHP 采用自动创建文件夹的机制，当用户布置好 ThinkPHP 的核心类库后，编写运行入口文件，则相关应用到的项目目录就会自动生成）。

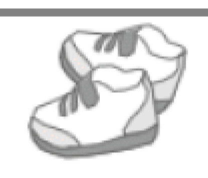




表 16.2　项目目录

目 录 名 称	主 要 作 用
index.php	项目入口文件
Common	项目公共目录，放置项目公共函数
Lang	项目语言包目录（可选）
Conf	项目配置目录，放置配置文件
Lib	项目基目录，通常包括Action和Model目录
Tpl	项目模板目录
Runtime	项目运行时的目录，包括Cache、Temp、Data和Log

16.2.2　自动生成目录

下面通过一个示例讲解在 ThinkPHP 框架中如何自动生成项目目录。

例 16.1　创建名称为 16.1 的项目，自动生成项目目录，其操作步骤如下。（**实例位置：配套资源\mr\16\example\16.1**）

（1）在网站根目录下创建文件夹，并将其命名为 16.1。

（2）将 ThinkPHP 核心类库存储于 16.1 目录下。

（3）编写入口文件 index.php，并将其存储于 16.1 目录下。index.php 文件代码如下：

```
<?php
define('THINK_PATH', 'ThinkPHP');              //定义ThinkPHP框架路径（相对于入口文件）
define('APP_NAME', '16.1');                    //定义项目名称
define('APP_PATH', '.');                       //定义项目路径
require(THINK_PATH."/ThinkPHP.php");           //加载框架入口文件
App::run();                                    //实例化一个网站应用实例
?>
```

在运行此文件之前，查看 16.1 项目的文件夹架构，如图 16.2 所示。

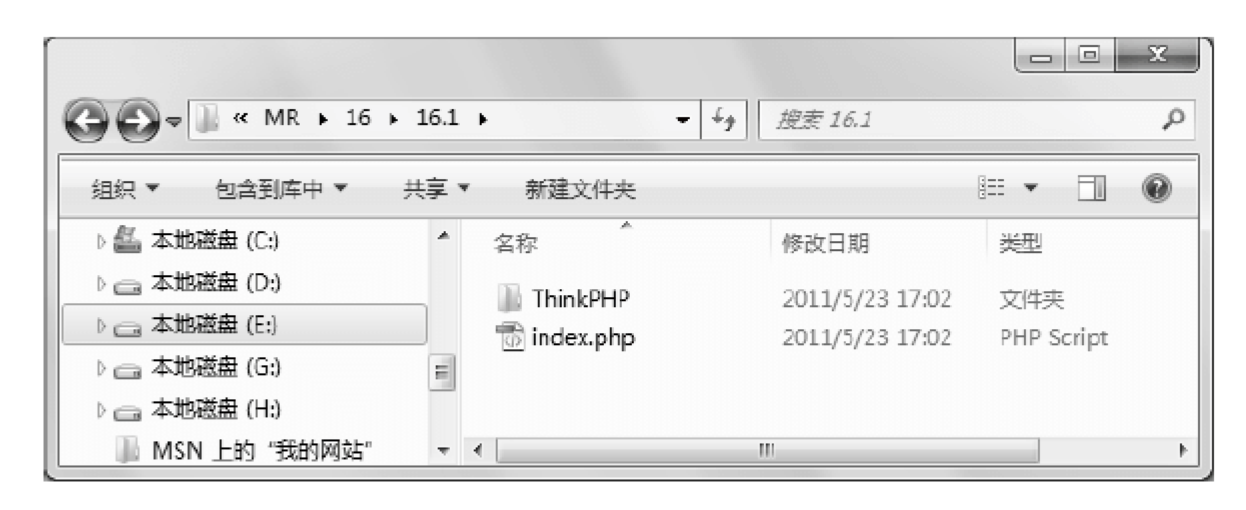


图 16.2　项目文件夹架构

在 IE 浏览器中运行此项目，将输出如图 16.3 所示的运行结果，此为 ThinkPHP 提供的测试内容。此时再次查看 16.1 项目文件夹，在项目根目录下将会自动生成项目目录，如图 16.4 所示。

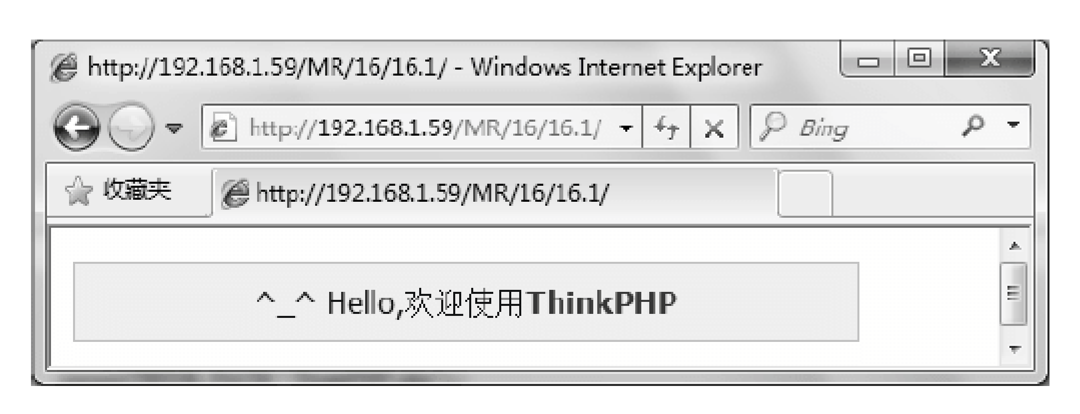


图 16.3　已连接到 ThinkPHP 框架


Note

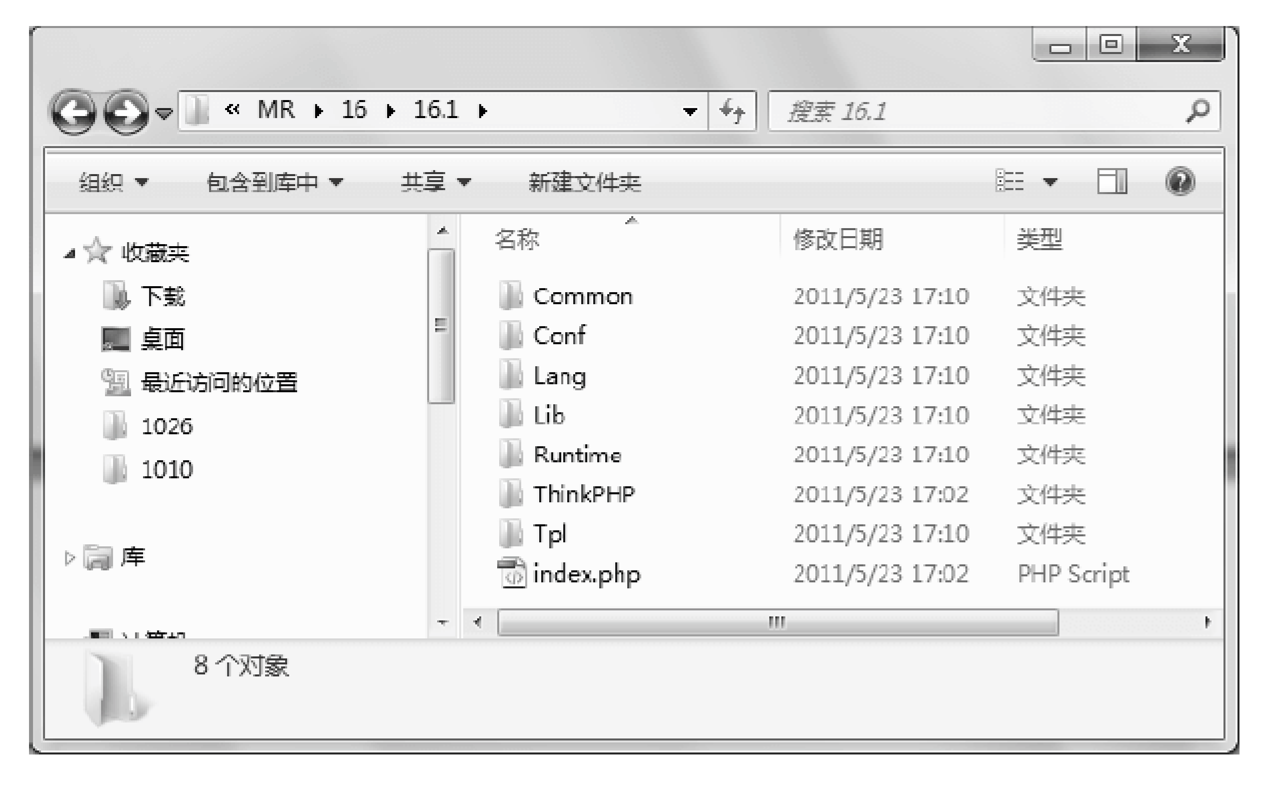


图 16.4　自动生成的项目目录

16.2.3　项目目录部署方案

在实际开发过程中，目录结构往往由于项目的复杂而变得复杂。这里向大家推荐两套标准的目录部署方案：方案一如图 16.5 所示；方案二采用分组模块，如图 16.6 所示。



图 16.5　项目部署方案一



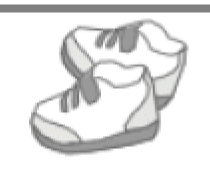
图 16.6　项目部署方案二

这样部署的好处是系统目录和项目目录可以存储于非 Web 访问目录下，网站目录下只需放置 Public 公共目录和 index.php 入口文件（如果是多个项目，则每个项目的入口文件都需要放到 Web 目录下），从而提高网站的安全性。

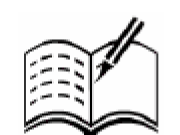
16.2.4　命名规范

ThinkPHP 框架有其自身的一定规范，要应用 ThinkPHP 框架开发项目，就要尽量遵守其规范。下面介绍一下 ThinkPHP 的命名规范。

☑ 类文件都是以.class.php 为后缀（这里指的是 ThinkPHP 内部使用的类库文件，不代表外部加载的类库文件），使用驼峰法命名，并且首字母大写，如 DbMysql.class.php。
☑ 函数、配置文件等其他类库文件之外的文件一般是以.php 为后缀（第三方引入的不做要求）。
☑ 确保文件的命名和调用大小写一致，这是因为在类 UNIX 系统中，对大小写是敏感的（而 ThinkPHP 在调试模式下，即使在 Windows 平台也会严格检查大小写）。
☑ 类名和文件名一致（包括上面说的大小写一致），如 UserAction 类的文件名是 UserAction.class.php，InfoModel 类的文件名是 InfoModel.class.php。
☑ 函数的命名使用小写字母和下划线的方式，如 get_client_ip。



- ☑ Action 控制器类以 Action 为后缀，如 UserAction、InfoAction。
- ☑ 模型类以 Model 为后缀，如 UserModel、InfoModel。
- ☑ 方法的命名使用驼峰法，并且首字母小写，如 getUserName。
- ☑ 属性的命名使用驼峰法，并且首字母小写，如 tableName。
- ☑ 以双下划线“__”开始的函数或方法作为魔法方法，如__call 和__autoload。
- ☑ 常量以大写字母和下划线命名，如 HAS_ONE 和 MANY_TO_MANY。
- ☑ 配置参数以大写字母和下划线命名，如 HTML_CACHE_ON。
- ☑ 语言变量以大写字母和下划线命名，如 MY_LANG。以下划线开头的语言变量通常用于系统语言变量，如_CLASS_NOT_EXIST_。
- ☑ 数据表和字段采用小写加下划线方式命名，如 think_user 和 user_name。



脚下留神：

在 ThinkPHP 中有一个函数命名的特例，就是单字母大写函数，这类函数通常是某些操作的快捷定义或者有特殊的作用，如 ADSL 方法等，它们有着特殊的含义。另外需要注意的是，ThinkPHP 默认使用 UTF-8 编码，所以应确保程序文件采用 UTF-8 编码格式保存，并且去掉 BOM 信息头（去掉 BOM 头信息有很多方式，不同的编辑器有各自的设置方法，另外也可以用工具进行统一检测和处理）。

16.2.5 项目构建流程

ThinkPHP 具有项目目录自动创建功能，因此构建项目应用程序非常简单，用户只需定义好项目的入口文件，在第一次访问入口文件时，系统会自动根据在入口文件中所定义的目录路径，迅速创建好项目的相关目录结构。在完成项目目录结构的创建后，再进行其他工作，图 16.7 展示了 ThinkPHP 创建项目的基本流程。

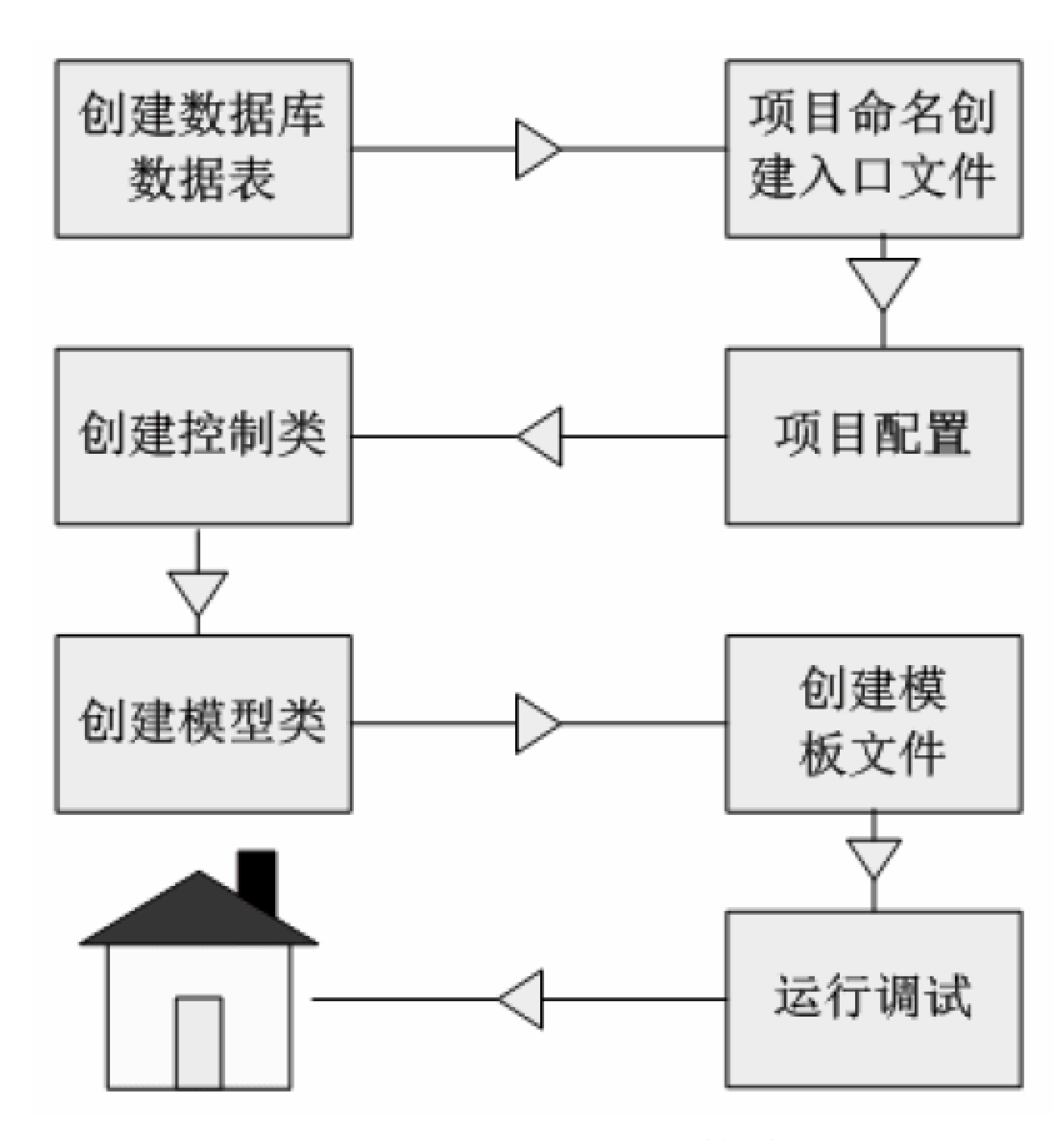
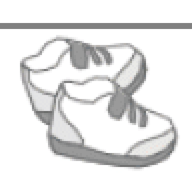


图 16.7 ThinkPHP 项目构建流程

例 16.2 根据图 16.7 所示流程，创建一个名称为 16.2 的项目，读取 db_database16 数据库



中的数据，其操作步骤如下。**（实例位置：配套资源\mr\16\example\16.2）**

（1）创建 db_database16 数据库，创建 think_user 数据表。数据表结构如图 16.8 所示。



服务器: localhost ▸ 数据库: db_database16 ▸ 表: think_user

字段	类型	整理	属性	空	默认	额外
id	int(10)			否	无	auto_increment
user	varchar(80)	utf8_unicode_ci		否	无	
pass	varchar(80)	utf8_unicode_ci		否	无	
address	varchar(80)	utf8_unicode_ci		否	无	

图 16.8　数据表结构

（2）载入 ThinkPHP 系统文件，编辑入口文件 index.php，创建名称为 16.2 的项目。index.php 的代码如下：

```
<?php
define('THINK_PATH', '../ThinkPHP/');          //定义 ThinkPHP 框架路径
define('APP_NAME', '16.2');                    //定义项目名称和路径
define('APP_PATH', '.');                       //定义项目名称和路径
require(THINK_PATH."/ThinkPHP.php");           //加载框架入口文件
App::run();                                    //实例化一个网站应用实例
?>
```

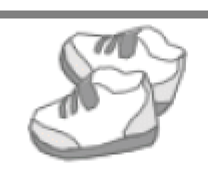
（3）自动生成的项目目录中已经创建了一个空的项目配置文件，位于项目的 Conf 目录下，名称是 config.php。重新编辑此文件，完成数据库的配置。config.php 文件的代码如下：

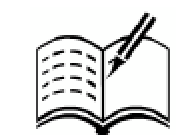
```
<?php
return array(
    'APP_DEBUG' => true,                       //开启调试模式
    'DB_TYPE'=> 'mysql',                       //数据库类型
    'DB_HOST'=> 'localhost',                   //数据库服务器地址
    'DB_NAME'=>'demo',                         //数据库名称
    'DB_USER'=>'root',                         //数据库用户名
    'DB_PWD'=>'111',                           //数据库密码
    'DB_PORT'=>'3306',                         //数据库端口
    'DB_PREFIX'=>'think_',                     //数据表前缀
);
?>
```

（4）在项目的 Lib\Action 目录下，定位到自动生成的 IndexAction.class.php 文件，这是 ThinkPHP 的控制器，即 Index 模块。重新编辑控制器的 index 方法，查询指定数据表中的数据，并且完成数据的循环输出。其代码如下：

```
<?php
class IndexAction extends Action{
    public function index() {
        $db = new Model('user');               //实例化模型类，参数数据表名称，不包含前缀
        $select = $db->select();               //查询数据
        $this->assign('select',$select);       //模板变量赋值
        $this->display();                      //输出模板
    }
}
?>
```

（5）在项目的 Tpl\default 目录下创建 Index 目录，存储 Index 模块的模板文件 index.html。





完成数据库中数据的循环输出，其代码如下：

```
<!--循环输出查询结果数据集-->
<volist name='select' id='user' >
ID:{$user.id}<br/>
用户名：  {$user.user}<br/>
地址：  {$user.address}<hr>
</volist>
```

（6）在 IE 浏览器地址栏中输入 http://127.0.0.1/MR/16/16.2/，其运行结果如图 16.9 所示。

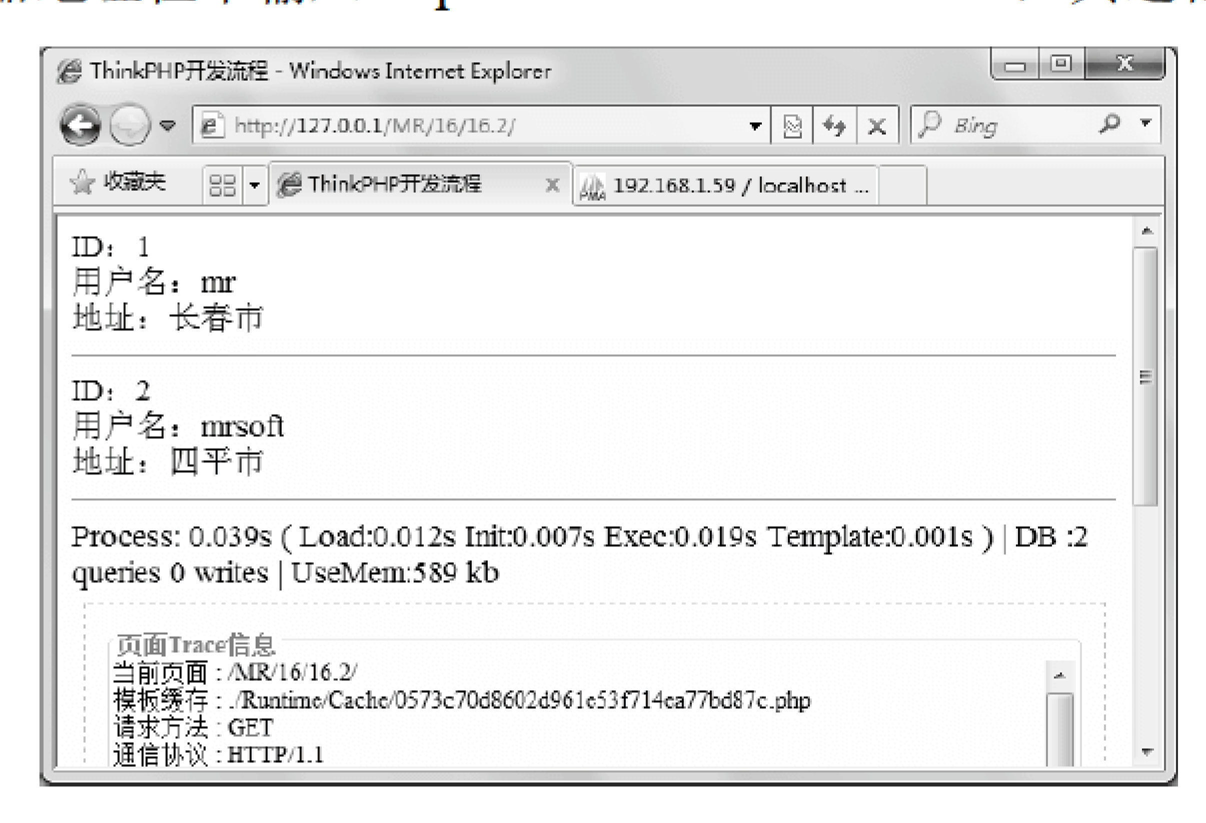


图 16.9　了解 ThinkPHP 项目的构建流程

☎小测试

1．ThinkPHP 框架中目录大体上可以分为哪两部分？
2．ThinkPHP 中内部类库文件的命名规则是什么？请举例说明。
3．简述通过 ThinkPHP 框架开发项目的流程。

16.3　ThinkPHP 的配置

视频讲解：配套资源\mr\16\video\ThinkPHP 的配置.exe

配置文件是 ThinkPHP 框架程序得以运行的基础条件，框架的很多功能都需要在配置文件中配置之后才可以生效，包括 URL 路由功能、页面伪静态和静态化等。ThinkPHP 提供了灵活的全局配置功能，采用最有效率的 PHP 返回数组方式定义，支持惯例配置、项目配置、调试配置和模块配置，并且会自动生成配置缓存文件，无需重复解析。

ThinkPHP 在项目配置上面创造了自己独有的分层配置模式，其配置顺序如图 16.10 所示。

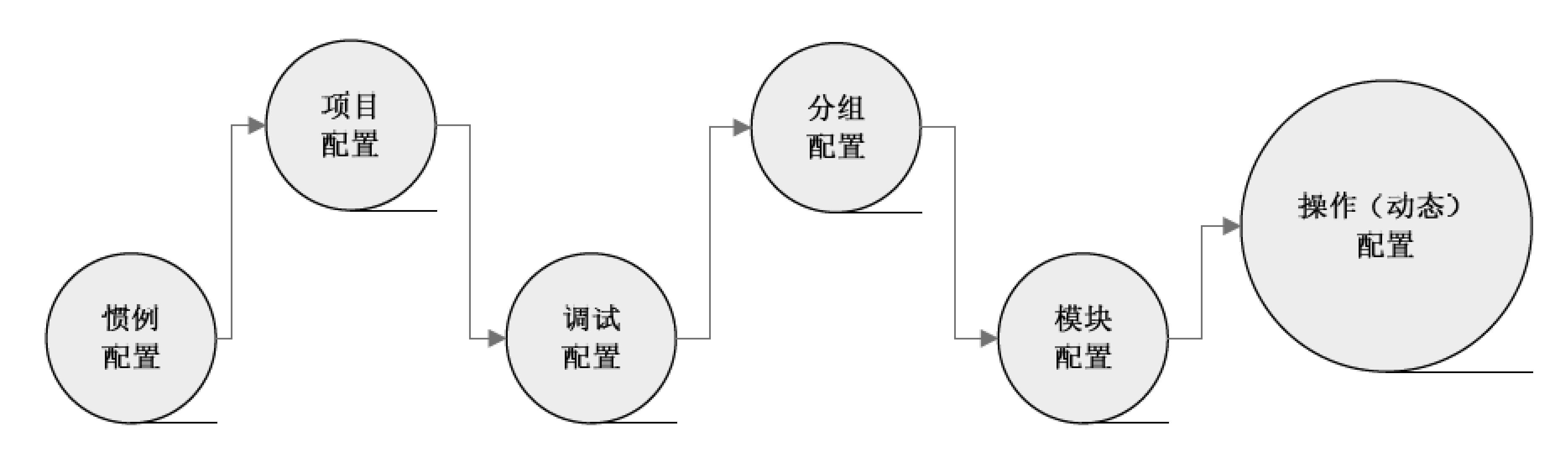
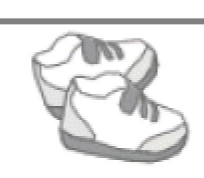


图 16.10　分层配置模式的顺序



以上是配置文件的加载顺序，但是因为后面的配置会覆盖之前的配置（在没有生效的前提下），所以优先顺序从右到左。系统的配置参数是通过静态变量全局存取的，存取方式非常简单高效。



Note

16.3.1 配置格式

ThinkPHP 框架中所有配置文件的定义格式均采用返回 PHP 数组的方式，格式为：

```
<?php
return array(
    'APP_DEBUG' => true,
    'URL_MODEL' => 2,
    //……更多的配置参数
);
?>
```

指点迷津：

配置参数不区分大小写（因为无论使用大小写定义都会转换成小写），但是习惯上保持大写定义的原则。另外，还可以在配置文件中使用二维数组来配置更多的信息。例如：

```
<?php
return array(
    'APP_DEBUG' => true,
    'USER_CONFIG' => array(
                'USER_AUTH' => true,
                'USER_TYPE' => 2,
    ),
);
?>
```

系统目前最多支持二维数组的配置级别，每个项目配置文件除了定义 ThinkPHP 所需要的配置参数之外，开发人员可以在其中添加项目需要的一些配置参数，用于自己的应用。项目配置文件默认存储于项目的 Conf 目录下。例如，在例 16.2 中，连接数据库的配置文件存储于项目的 16.2\Conf\config.php 文件中。

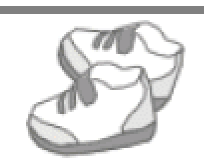
多学两招：

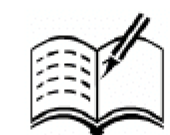
项目配置指的是项目的全局配置，因为一个项目除了可以定义项目配置文件之外，还可以定义模块配置文件，用于针对某个特定的模块进行特殊的配置。它们的定义格式都是一致的，区别只是配置文件命名的不同。系统会自动在不同的阶段读取配置文件。

16.3.2 调试配置

如果启用调试模式，则会导入框架默认的调试配置文件，该文件位于 Think\Common\debug.php 中，如果没有检测到项目的调试配置文件，就会直接使用默认的调试配置参数。项目定义自身的调试配置文件，则会和默认的调试配置文件合并，也就是说，项目配置文件也只需要配置和默认调试配置不同的参数或新增的参数。

调试配置文件也位于项目配置目录下，文件名是 debug.php。通常情况下，在调试配置文





件中可以进行一些开发模式所需要的配置。例如，配置额外的数据库连接用于调试、开启日志写入便于查找错误信息、开启页面 Trace 输出更多的调试信息等。系统默认的调试配置文件中设置如下内容：

- ☑ 开启日志记录。
- ☑ 关闭模板缓存。
- ☑ 记录 SQL 日志。
- ☑ 关闭字段缓存。
- ☑ 开启运行时间详细显示（包括内存、缓存情况）。
- ☑ 开启页面 Trace 信息显示。
- ☑ 严格检查文件大小写（即使是 Windows 平台）。

☎小测试

1．创建一个数据库连接的配置文件。
2．创建一个以 PDO 方法操作数据库的配置文件。

16.4 ThinkPHP 的控制器

视频讲解：配套资源\mr\16\video\ThinkPHP 的控制器.exe

16.4.1 控制器

ThinkPHP 的控制器就是模块类，通常位于项目的 Lib\Action 目录下。类名是模块名加上 Action 后缀，如 IndexAction 类表示 Index 模块。控制器类必须继承系统的 Action 基础类，这样才能确保使用 Action 类内置的方法。

而 index 操作其实就是 IndexAction 类的一个公共方法，所以在浏览器中输入 http://localhost/myApp/index.php/Index/index/，其实就是执行 IndexAction 类的 index（公共）方法。

例 16.3 对自动生成的项目目录中的控制器进行修改，使其输出自己编译的内容，其操作步骤如下。**（实例位置：配套资源\mr\16\example\16.3）**

（1）创建名称为 16.3 的项目，将 ThinkPHP 核心类库存储于 16.3 目录下。

（2）编写入口文件 index.php，将其存储于 16.3 目录下。index.php 文件的代码如下：

```
<?php
define('THINK_PATH', '../ThinkPHP');                //定义 ThinkPHP 框架路径（相对于入口文件）
define('APP_NAME', '16.3');                         //定义项目名称
define('APP_PATH', '.');                            //定义项目路径
require(THINK_PATH."/ThinkPHP.php");                //加载框架入口文件
App::run();                                         //实例化一个网站应用实例
?>
```

（3）运行 index.php 文件，在 16.3 目录下自动创建项目目录，其运行效果如图 16.11 所示。

^_^ Hello,欢迎使用ThinkPHP

图 16.11 自动创建项目目录





Note

（4）在自动创建的项目目录中，控制器 IndexAction 中输出的是 ThinkPHP 设置的内容，此处对该内容进行修改，输出“明日科技欢迎您！”。修改后 16.3\Lib\Action\IndexAction.class.php 文件的代码如下：

```
<?php
class IndexAction extends Action{
    public function index(){
        header("Content-Type:text/html; charset=utf-8");        //设置编码格式
        echo "<div style='font-weight:normal;color:blue;float:left;width:345px;text-align:center;border:1px solid silver;background:#E8EFFF;padding:8px;font-size:14px;font-family:Tahoma'>^_^ <span style='font-weight:bold;color:red'>明日科技欢迎您！</span></div>";        //输出内容
    }
}
?>
```

在对控制器的内容进行修改后，重新运行项目，在 IE 浏览器中输入 http://192.168.1.59/MR/16/16.3/，将输出如图 16.12 所示的效果。

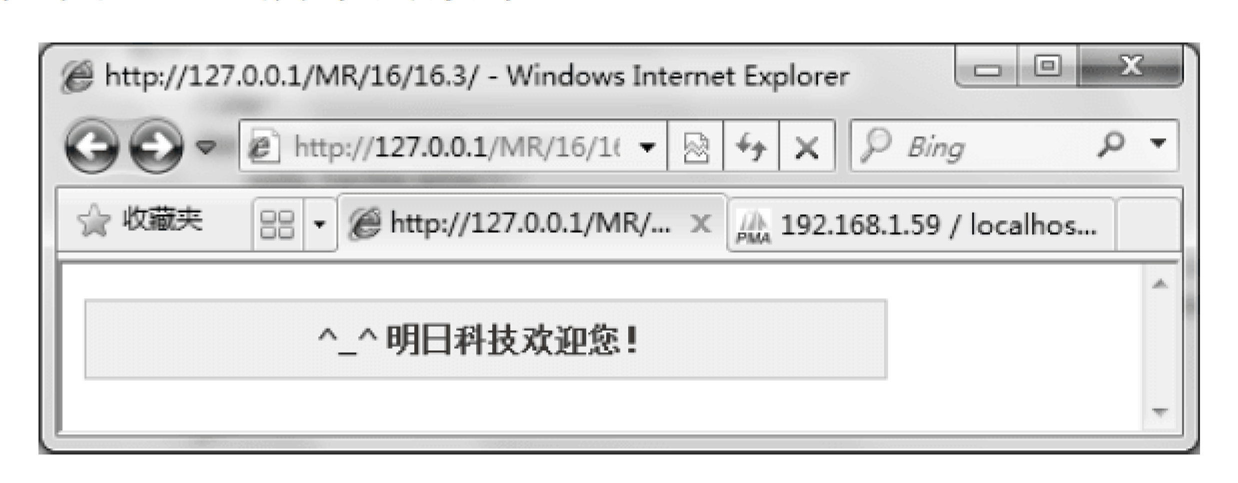


图 16.12　输出控制器中的内容

指点迷津：

每个模块的操作并非一定要定义操作方法，如果只是希望输出一个模板，既没有变量也没有任何的业务逻辑，那么只要按照规则定义好操作对应的模板文件即可，而不需要定义操作方法。例如，在 IndexAction 中如果没有定义 help 方法，但是存在对应的 Index/help.html 模板文件，那么 http://localhost/16/index.php/Index/help/依然可以正常运行，因为系统找不到 IndexAction 类的 help 方法，会自动定位到 Index 模块的模板目录中查找 help.html 模板文件，然后直接输出。

16.4.2　跨模块调用

在开发过程中，经常会在当前模块调用其他模块的方法，此时就涉及到跨模块调用。下面通过 A 和 R 两个快捷方法完成跨模块调用。

```
$User = A("User");                //实例化 UserAction 控制器对象
$User->insert();                  //调用 User 模块的 importUser 操作方法
```

这里的 A("User") 是一个快捷方法，与下面的代码等效：

```
import("@.Action.UserAction");
$User = new UserAction();
```

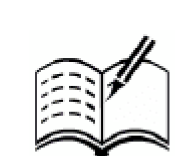
事实上，在这个例子中，还有比 A 方法更简单的调用方法，例如：

```
R("User","insert");               //远程调用 UserAction 控制器的 insert 操作方法
```

上面只是在当前项目中调用，如果需要在多个项目之间调用方法，一样可以完成，如下所示：



```
$User = A("User","Admin");          //实例化 Admin 项目的 UserAction 控制器对象
$User->insert();                    //调用 Admin 项目 UserAction 控制器的 insert 操作方法
 R("User","insert","Admin");        //远程调用 Admin 项目的 UserAction 控制器的 insert 操作方法
```



例 16.4 应用跨模块调用的方法，在前台控制器中调用后台项目中的 insert 方法完成用户信息的添加操作，其操作步骤如下。**（实例位置：配套资源\mr\16\example\16.4）**

（1）创建 16.4 项目根目录，在根目录下分别创建前台项目文件夹 Home、后台项目文件夹 Admin 和 Public 文件夹存储 CSS、图片和 JS 脚本等文件。

（2）在 16.4 项目根目录下，编辑 index.php 前台入口文件和 admin.php 后台入口文件。其关键代码如下：

```
//Index.php 前台入口文件
<?php
define('THINK_PATH', '../ThinkPHP');                //定义 ThinkPHP 框架路径（相对于入口文件）
define('APP_NAME', 'Home');                         //定义项目名称
define('APP_PATH', './Home');                       //定义项目路径
require(THINK_PATH."/ThinkPHP.php");                //加载框架入口文件
App::run();                                         //实例化一个网站应用实例
?>
//Admin.php 后台入口文件
<?php
define('THINK_PATH', '../ThinkPHP');                //定义 ThinkPHP 框架路径（相对于入口文件）
define('APP_NAME', 'Admin');                        //定义项目名称
define('APP_PATH', './Admin');                      //定义项目路径
require(THINK_PATH."/ThinkPHP.php");                //加载框架入口文件
App::run();                                         //实例化一个网站应用实例
?>
```

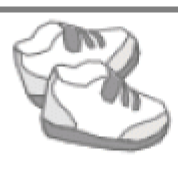
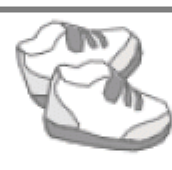
（3）在 IE 浏览器中运行前后台的入口文件，自动创建项目目录。

（4）定位到 Admin\Conf 目录下，编辑 config.php 文件，完成后台项目中数据库的配置。其代码如下：

```
<?php
return array(
    'APP_DEBUG' => false,                           //关闭调试模式
    'DB_TYPE'=> 'mysql',                            //数据库类型
    'DB_HOST'=> 'localhost',                        //数据库服务器地址
    'DB_NAME'=>'db_database16',                     //数据库名称
    'DB_USER'=>'root',                              //数据库用户名
    'DB_PWD'=>'111',                                //数据库密码
    'DB_PORT'=>'3306',                              //数据库端口
    'DB_PREFIX'=>'think_',                          //数据表前缀
);
?>
```

（5）定位到 Admin\Lib\Action 目录下，编写后台项目的控制器。首先创建 Index 模块，继承系统的 Action 基础类，定义 index 方法读取指定数据表中的数据，并且将查询结果赋给模板变量，最终指定模板页。IndexAction.class.php 的代码如下：

```
<?php
header("Content-Type:text/html; charset=utf-8");   //设置页面编码格式
```

```
class IndexAction extends Action{
    public function index() {
        $db = new Model('user');        //实例化模型类，参数数据表名称，不包含前缀
        $select = $db->select();        //查询数据
        $this->assign('select',$select); //模板变量赋值
        $this->display();               //输出模板
    }
}
?>
```

然后创建 User 模块，同样继承系统的 Action 基础类，定义 insert 方法，实例化模型类，将表单中提交的数据添加到指定的数据表中，添加成功后重定向到后台主页。UserAction.class.php 的代码如下：

```
<?php
header("Content-Type:text/html; charset=utf-8");        //设置页面编码格式
class UserAction extends Action{                         //定义类，继承基础类
    public function insert() {                           //定义方法
        $ins = new Model('user');     //实例化模型类，传递参数为没有前缀的数据表名称
        $ins->Create();                                  //创建数据对象
        $result = $ins->add();                           //写入数据库
        $this->redirect('Index/index','', 5,'页面跳转中'); //页面重定向
    }
}
?>
```

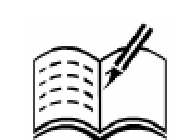
（6）定位到 Admin\Tpl\default 目录下，首先创建 Index 模块文件夹，编辑 index 操作的模板文件 index.html，循环输出模板变量传递的数据。index.html 模板文件的关键代码如下：

```
<volist name='select' id='user' >
  <tr class="content">
    <td bgcolor="#FFFFFF"> {$user.id}</td>
    <td bgcolor="#FFFFFF"> {$user.user}</td>
    <td bgcolor="#FFFFFF"> {$user.address}</td>
  </tr>
  </volist>
```

然后创建 User 模块文件夹，编程 insert 操作的模板文件 index.html，创建添加用户信息的表单。其关键代码如下：

```
<form method="post"   action="__URL__/insert" >
<table width="265" border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td class="title" id="td">用户名：</td>
        <td><input name="user" type="text" size="15" /></td>
      </tr>
      <tr>
        <td class="title" id="td">密码：</td>
        <td><input name="pass" type="password" size="15" /></td>
      </tr>
      <tr>
```






```
                <td class="title" id="td">地址：</td>
                <td><input name="address" type="text" size="20" /></td>
            </tr>
        </table>
    <input type="image" name="imageField" id="imageField" src="__ROOT__/Public/images/66_05.gif" />
    </form>
```

（7）定位到 Home\Conf 目录，编辑前台项目的数据库配置文件 config.php。

（8）定位到 Home\Lib\Action 目录，创建前台项目控制器 Index。定义 index 方法查询数据库中的用户信息，并且将查询结果赋给模板变量；定义 insert 方法，通过 R 快捷方式调用 Admin 项目 UserAction 控制器的 insert 操作方法完成数据的添加操作。IndexAction.class.php 的代码如下：

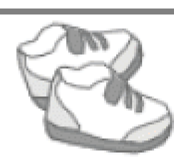
```
<?php
header("Content-Type:text/html; charset=utf-8");      //设置页面编码格式
class IndexAction extends Action{
    public function index() {
        $db = new Model('user');                    //实例化模型类，参数数据表名称，不包含前缀
        $select = $db->select();                      //查询数据
        $this->assign('select',$select);              //模板变量赋值
        $this->display();                             //输出模板
    }
    public function insert() {
        $ins=R("User","insert","Admin");//远程调用 Admin 项目 UserAction 控制器的 insert 操作方法
        $ins->Create();                               //创建数据对象
        $result = $ins->add();                        //写入数据库
    }
}
?>
```

（9）定位到 Home\Tpl\default 目录下，创建 Index 模块文件夹，编辑 index 操作的模板文件 index.html。在 index.html 模板文件中创建表单提交用户注册信息，循环输出模板变量传递的数据。

（10）在 IE 浏览器中输入 http://127.0.0.1/mr/16/example/16.4/，其运行效果如图 16.13 所示。



图 16.13　跨模块调用完成用户注册






多学两招:

网页重定向

Action 类的 redirect 方法可以实现页面的重定向功能。该方法的定义规则如下（方括号内参数根据实际应用决定）:

redirect('[项目://][路由@][分组名-模块/]操作? 参数 1=值 1[&参数 N=值 N]')

或者用数组的方式传入参数:

redirect('[项目://][路由@][分组名-模块/]操作',array('参数 1'=>'值 1' [,'参数 N'=>'值 N']))

如果不定义项目和模块，则表示当前项目和模块名称。例如:

```
$this->redirect('Index/index','', 5,'页面跳转中');      //页面重定向
```

停留 5 秒后跳转到 Index 模块的 index 操作，并且显示页面跳转中字样，重定向后将改变当前的 URL 地址。

16.5 ThinkPHP 的模型

视频讲解：配套资源\mr\16\video\ThinkPHP 的模型.exe

顾名思义，模型就是按照某一个形状进行操作的代名词，其主要作用是封装数据库的相关逻辑。也就是说，每执行一次数据库操作，都要遵循定义的数据模型规则来完成。

16.5.1 模型的命名

在定义模型时，ThinkPHP 要求数据库的表名和模型类的命名遵循一定的规范，首先数据库的表名和字段全部采用小写形式，模型类的命名规则是除去表前缀的数据表名称，并且首字母大写，然后加上模型类的后缀定义。

例如，UserModel 表示 User 数据对象，假设数据库的前缀定义是 think_，则其对应的数据表是 think_user；UserTypeModel 对应的数据表是 think_user_type。

如果用户的规则和系统的约定不符合，那么需要设置 Model 类的 tableName 属性。在 ThinkPHP 的模型中有两个数据表名称的定义:

1. tableName

不包含表前后缀的数据表名称，一般情况下默认和模型名称相同，只有当表名和当前的模型类的名称不相同时才需要定义。例如，在数据库中有一个 think_categories 表，而定义的模型类名称是 CategoryModel，按照系统的约定，该模型的名称是 Category，对应的数据表名称应是 think_category（全部小写），但是现在的数据表名称是 think_categories，因此就需要设置 tableName 属性来改变默认的规则（假设已经在配置文件中定义了 DB_PREFIX 为 think_）。

```
protected $tableName = 'categories';
```

脚下留神:

该属性的定义不需要加表的前缀 think_。

2. trueTableName

包含前后缀的数据表名称，也就是数据库中的实际表名，该名称无需设置，只有当上面的规则都不适用或特殊情况下才需要设置。例如，数据库中有一个表（top_depts）的前缀和其他

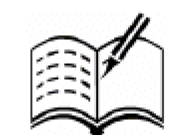


表的前缀不同（不是 think_而是 top_），此时就需要定义 trueTableName 属性：

```
protected $trueTableName = 'top_depts';
```

脚下留神：

trueTableName 需要完整的表名定义。



除了数据表的定义外，还可以对数据库进行定义：

dbName 定义模型当前对应的数据库名称，只有当前的模型类对应的数据库名称和配置文件不相同时才需要定义，例如：

```
protected $dbName = 'top';
```

16.5.2 实例化模型

在 ThinkPHP 2.0 版本中，无需进行任何模型定义（只有在需要封装单独的业务逻辑时模型类才是必须被定义的），即可直接进行模型的实例化操作。根据不同的模型定义，实例化模型的方法也有所不同，下面来分析一下在什么情况下使用什么方法。

1. 实例化基础模型（Model）类

在未定义任何模型时，可以使用下面的方法实例化一个模型类来进行操作：

```
$User = new Model('User');
$User->select();          //进行其他的数据操作
```

或者使用 M 快捷方法进行实例化，其效果是相同的。

```
$User = M('User');
$User->select();          //进行其他的数据操作
```

这种方法最简单高效，因为不需要定义任何的模型类，所以支持跨项目调用。其缺点也是因为没有自定义的模型类，因此无法写入相关的业务逻辑，只能完成基本的 CURD 操作。在例 16.2 和例 16.4 中采用的都是实例化基础模型类，对数据库中数据进行读取、添加操作。

2. 实例化其他模型类

第一种方式实例化因为没有模型类的定义，因此很难封装一些额外的逻辑方法，不过大多数情况下，也许只是需要扩展一些通用的逻辑，那么就可以尝试下面一种方法。

M 方法默认是实例化 Model 类，如果需要实例化其他模型类，可以使用：

```
$User = M('User', 'CommonModel');
```

上面的方法等效于：

```
$User = new CommonModel('User');
```

因为系统的模型类都能够自动加载，因此不需要在实例化之前手动进行类库导入操作。模型类 CommonModel 必须继承 Model，如果没有定义别名导入，则需要放在项目 Model 下。我们可以在 CommonModel 类中定义一些通用的逻辑方法，就可以省去为每个数据表定义具体的模型类，如果项目的数据表超过 100 个，而且大多数都是执行基本的 CURD 操作，只是个别模型有一些复杂的业务逻辑需要封装，那么第一种方式和第二种方式的结合是一个不错的选择。

3. 实例化用户定义的模型（×××Model）类

这种情况是使用最多的，一个项目不可避免地需要定义自身的业务逻辑实现，就需要针对





每个数据表定义一个模型类，如 UserModel、InfoModel 等。

定义的模型类通常都是放到项目的 Lib\Model 目录下。例如：

```
class UserModel extends Model{
    Public function myfun(){
        //添加自己的业务逻辑
        //………
    }
}
```

其实模型类还可以继承一个用户自定义的公共模型类，而不是只能继承 Model 类。要实例化自定义模型类，可以使用下面的方式：

```
$User = new UserModel();
$User->select();                    //进行其他的数据操作
```

另外，还可以使用 D 快捷方法进行实例化，其效果是相同的。

```
$User = D('User');
$User->select();                    //进行其他的数据操作
```

D 方法可以自动检测模型类，不存在时系统会抛出异常，同时对于已实例化过的模型，不会重复实例化。默认的 D 方法只能支持调用当前项目的模型，如果需要跨项目调用，需要使用：

```
$User = D('User', 'Admin');         //实例化 Admin 项目下面的 User 模型
$User->select();
```

如果启用模块分组功能，还可以使用：

```
$User = D('Admin.User');
```

4. 实例化空模型类

如果仅仅是使用原生 SQL 查询，则不需要使用额外的模型类，实例化一个空模型类即可进行操作，例如：

```
$Model = new Model();
```

或者使用 M 快捷方法实例化是等效的。

```
// $Model = M();
$Model->query('SELECT * FROM think_user where status=1');
```

空模型类也支持跨项目调用。

例 16.5 通过 M 方法实例化 Model 类，完成数据库中用户信息和类别信息的输出，其关键操作步骤如下。（**实例位置：配套资源\mr\16\example\16.5**）

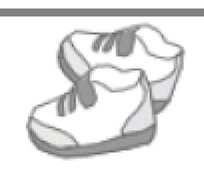
（1）创建 16.5 项目根目录，在根目录下创建项目文件夹 App 和 Public 文件夹存储 CSS、图片和 JS 脚本等文件。

（2）在 16.5 项目根目录下编辑 index.php 入口文件。其关键代码如下：

```
<?php
define('THINK_PATH', '../ThinkPHP');         //定义 ThinkPHP 框架路径（相对于入口文件）
define('APP_NAME', '16.5');                  //定义项目名称
define('APP_PATH', './App');                 //定义项目路径
require(THINK_PATH."/ThinkPHP.php");         //加载框架入口文件
App::run();                                  //实例化一个网站应用实例
?>
```

（3）在 IE 浏览器中运行入口文件，自动生成项目目录。

（4）定位到 App\Conf 目录下，编辑 config.php 文件，完成项目中数据库的配置。其代码





如下：

```
<?php
return array(
  'APP_DEBUG' => false,                    //关闭调试模式
  'DB_TYPE'=> 'mysql',                     //数据库类型
  'DB_HOST'=> 'localhost',                 //数据库服务器地址
  'DB_NAME'=>'db_database16',              //数据库名称
  'DB_USER'=>'root',                       //数据库用户名
  'DB_PWD'=>'111',                         //数据库密码
  'DB_PORT'=>'3306',                       //数据库端口
  'DB_PREFIX'=>'think_',                   //数据表前缀
);
?>
```

（5）定位到 App\Lib\Action 目录下，编写项目的控制器。创建 Index 模块，继承系统的 Action 基础类，定义 index 方法，通过 M 方法实例化模型类，读取 think_user 数据表中的数据，并且将查询结果赋给模板变量，指定模板页；定义 type 方法，通过 M 方法实例化模型类，读取类型数据表 think_type 中的数据，同样将查询结果赋给模板变量，指定模板页。IndexAction.class.php 的代码如下：

```
<?php
header("Content-Type:text/html; charset=utf-8"); //设置页面编码格式
class IndexAction extends Action{
    public function index(){
        $db = M('User');                          //实例化模型类，参数数据表名称，不包含前缀
        $select = $db->select();                  //查询数据
        $this->assign('select',$select);          //模板变量赋值
        $this->display();                         //指定模板页
    }
    public function type(){
        $dba = M('Type');                         //实例化模型类，参数数据表名称，不包含前缀
        $select = $dba->select();                 //查询数据
        $this->assign('select',$select);          //模板变量赋值
        $this->display('type');                   //指定模板页
    }
}
?>
```

（6）定位到 App\Tpl\default 目录下，创建 Index 模块文件夹。首先，编辑 index 操作的模板文件 index.html，循环输出模板变量传递的数据。其关键代码如下：

```
<volist name='select' id='user' >
  <tr class="content">
    <td bgcolor="#FFFFFF"> {$user.id}</td>
    <td bgcolor="#FFFFFF"> {$user.user}</td>
    <td bgcolor="#FFFFFF"> {$user.address}</td>
  </tr>
  </volist>
```




然后编辑 type.html 模板文件，循环输出类型数据表中的数据。其关键代码如下：

```
<volist name='select' id='type' >
<tr class="content">
  <td bgcolor="#FFFFFF"> {$type.id}</td>
  <td bgcolor="#FFFFFF"> {$type.typename}</td>
  <td bgcolor="#FFFFFF"> {$type.dates}</td>
</tr>
</volist>
```

（7）在 IE 浏览器中输入 http://127.0.0.1/mr/16/example/16.5/，其运行效果如图 16.14 所示。在 IE 浏览器中输入 http://127.0.0.1/mr/16/example/16.5/index.php/Index/type，其运行效果如图 16.15 所示。

用户信息		
ID	名称	地址
1	mr	长春市
2	mrsoft	四平市
3	Tsoft	长春市

图 16.14　输出用户信息

类别输出		
ID	类别名称	添加时间
1	PHP	2011-05-16
2	JAVA	2011-05-16
3	C#	2011-05-16
4	C++	2011-05-16

图 16.15　输出类别信息

多学两招：

在 Model 类中，根本没有定义任何 User 表、Type 表的字段信息，但是系统是如何做到属性对应数据表的字段呢？这是因为 ThinkPHP 可以在运行时自动获取数据表的字段信息（确切地说，是在第一次运行时将其存储于缓存文件，以后会永久缓存字段信息，除非设置不缓存或删除），包括数据表的主键字段和是否自动增长等，如果需要显式获取当前数据表的字段信息，可以使用模型类的 getDbFields 方法来获取。如果在开发过程中修改了数据表的字段信息，需要清空 Data/_fields 目录下面的缓存文件，让系统重新获取更新的数据表字段信息。

16.5.3　属性访问

ThinkPHP 利用 PHP5 的魔术方法机制来实现属性的直接访问。这也是最常用的访问方式，即通过数据对象访问，例如：

```
<?php
$User = new Model('User');
$User->find(1);
echo $User->name;              //获取 name 属性的值
$User->name = 'ThinkPHP';      //设置 name 属性的值
?>
```

还有一种属性的操作方式是通过返回数组的方式，例如：

```
<?php
$Type = D('Type');             //注意这里返回的 type 数据是一个数组
$type = $Type->find(1);
echo $type['name'];            //获取 type 属性的值
$type['name'] = 'ThinkPHP';    //设置 type 属性的值
?>
```



Note

16.5.4 连接数据库

ThinkPHP 内置抽象数据库访问层，把不同的数据库操作封装起来，只需使用公共的 Db 类进行操作，而无需针对不同的数据库写不同的操作代码，Db 类会自动调用相应的数据库适配器来处理。目前的数据库包括 MySQL、MsSQL、PgSQL、SQLite、Oracle、Ibase 及 PDO 的支持，如果应用需要使用数据库，则必须配置数据库连接信息，数据库的配置文件有多种定义方式：

（1）在项目配置文件中定义。在前面的实例中已经讲过。其代码如下：

```
<?php
return array(
    'APP_DEBUG' => false,              //关闭调试模式
    'DB_TYPE'=> 'mysql',               //数据库类型
    'DB_HOST'=> 'localhost',           //数据库服务器地址
    'DB_NAME'=>'db_database16',        //数据库名称
    'DB_USER'=>'root',                 //数据库用户名
    'DB_PWD'=>'111',                   //数据库密码
    'DB_PORT'=>'3306',                 //数据库端口
    'DB_PREFIX'=>'think_',             //数据表前缀
);
?>
```

系统推荐使用该方式，因为一般一个项目的数据库访问配置是相同的。该方法系统在连接数据库时会自动获取，无需手动连接。

指点迷津：

可以对每个项目定义不同的数据库连接信息，还可以在调试配置文件中定义调试数据库的配置信息，如果在项目配置文件和调试模式配置文件中同时定义了数据库连接信息，那么在调试模式下后者生效，部署模式下前者生效。

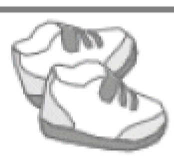
（2）使用 DSN 方式在初始化 Db 类时传参数，代码如下：

```
$db_dsn="mysql://root:111@127.0.0.1:3306/db_database16";    //定义 DSN
$db = new Db();                                             //执行类的实例化
$conn=$db->getInstance($db_dsn);                            //连接数据库，返回数据库驱动类
```

该方式主要用于在控制器中手动连接数据库的情况，或者用于创建多个数据库连接。

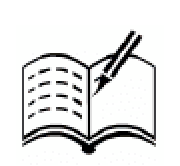
（3）使用数组传参数，代码如下：

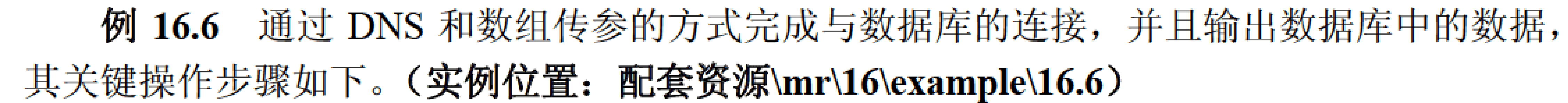
```
$dsn = array(
    'dbms'     => 'mysql',
    'username' => 'username',
    'password' => 'password',
    'hostname' => 'localhost',
    'hostport' => '3306',
    'database' => 'dbname'
);
```



```
    $db = new Db();
    $conn=$db->getInstance($dsn);                              //连接数据库，返回数据库驱动类
```

该方式用于手动连接数据库或创建多个数据库连接。





例 16.6 通过 DNS 和数组传参的方式完成与数据库的连接，并且输出数据库中的数据，其关键操作步骤如下。（**实例位置：配套资源\mr\16\example\16.6**）

本例是例 16.5 的延伸，仍然输出数据库中用户和类别表中的数据，只是对其连接数据库的方法进行了修改。

（1）删除 App\Conf 目录下的配置文件 config.php。

（2）在 App\Lib\Action\Index 目录下，修改控制器 Index。在 index 方法中，应用 DNS 方式完成与数据库的连接，并且查询 think_user 表中的数据。其关键代码如下：

```
public function index(){
    $db_dsn="mysql://root:111@127.0.0.1:3306/db_database16";  //定义 DSN
    $db = new Db();                                          //执行类的实例化
    $conn=$db->getInstance($db_dsn);                         //连接数据库，返回数据库驱动类
    $select=$conn->query('select * from think_user');        //执行查询语句
    $this->assign('select',$select);                         //模板变量赋值
    $this->display();                                        //指定模板页
}
```

（3）在 type 方法中，应用数组传递参数，完成数据库的连接操作，并且查询 think_type 表中的数据。其关键代码如下：

```
public function type(){
    $dsn = array(
        'dbms'      => 'mysql',
        'username' => 'root',
        'password' => '111',
        'hostname' => 'localhost',
        'hostport' => '3306',
        'database' => 'db_database16'
    );
    $db = new Db();
    $conn=$db->getInstance($dsn);                       //连接数据库，返回数据库驱动类
    $select=$conn->query('select * from think_type');  //执行查询语句
    $this->assign('select',$select);                    //模板变量赋值
    $this->display('type');                             //指定模板页
}
```

上述是在例 16.5 中所做的修改，至于其他步骤与例 16.5 相同，这里不再赘述。其运行结果也与例 16.5 相同。

（4）在模型类中定义参数，连接数据库，代码如下：

```
protected $connection = array(
    'dbms'      => 'mysql',
    'username' => 'username',
    'password' => 'password',
    'hostname' => 'localhost',
    'hostport' => '3306',
```



```
        'database' => 'dbname'
    );
```

或者使用下面的方式定义：

```
    protected $connection = "mysql://username:password@localhost:3306/DbName";
```

如果在某个模型类中定义了 connection 属性，则在实例化模型对象时，会使用该数据库连接信息进行数据库连接。通常用于某些数据表位于当前数据库连接之外的其他数据库。

指点迷津：

ThinkPHP 并不是在一开始就连接数据库，而是在有数据查询操作时才会去连接数据库。特殊的情况是，在系统第一次操作模型时，框架会自动连接数据库获取相关模型类的数据字段信息，并缓存下来。

（5）使用 PDO 方式连接数据库。这里在项目配置文件中，应用 PDO 连接数据库。其定义的数组内容如下：

```
    return array(
        'DB_TYPE'=> 'pdo',
        //注意 DSN 的配置针对不同的数据库有所区别
        'DB_DSN'=> 'mysql:host=localhost;dbname=db_database16',
        'DB_USER'=>'root',
        'DB_PWD'=>'111',
        'DB_PREFIX'=>'think_',
        //其他项目配置参数
        'APP_DEBUG' => false,              //关闭调试模式
    );
```

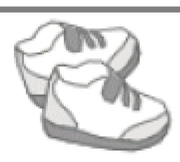
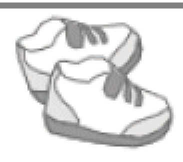
脚下留神：

在使用 PDO 方式时要注意检查 PHP 环境是否开启相关的 PDO 模块，同时还要确保用户的 ThinkPHP 核心包中包含 DbPdo.class.php 文件。另外，还要注意参数 DB_DSN 仅对 PDO 方式连接有效。

例 16.7 在项目配置文件中，以 PDO 方式连接数据库，并且输出数据库中的数据，其关键操作步骤如下。（**实例位置：配套资源\mr\16\example\16.7**）

在例 16.5 中，数据库的连接方法定义到配置文件 config.php 中，而应用 PDO 连接 MySQL 数据库仍然需要在配置文件中进行操作，那么此时只需对例 16.5 中的 config.php 文件进行修改，就完成了一个新的示例——例 16.7，应用 PDO 连接 MySQL 数据，并且输出查询结果。修改后的 config.php 文件的代码如下：

```
    <?php
    return array(
        'DB_TYPE'=> 'pdo',
        //注意 DSN 的配置针对不同的数据库有所区别
        'DB_DSN'=> 'mysql:host=localhost;dbname=db_database16',
        'DB_USER'=>'root',
        'DB_PWD'=>'111',
```

```
        'DB_PREFIX'=>'think_',
        //其他项目配置参数
        'APP_DEBUG' => false,                //关闭调试模式
    );
    ?>
```



本例的运行结果与例 16.5 相同，这里不再赘述。

16.5.5 创建数据

ThinkPHP 可以自动根据表单数据创建数据对象，该优势在一个数据表的字段非常多的情况下尤其明显。例如，在 User 控制器中，定义 insert 方法，首先实例化模型类，然后调用 Create 方法根据表单提交的 POST 数据创建数据对象，最后调用 add 方法把创建的数据对象写入数据库。其关键代码如下：

```
class UserAction extends Action{                              //定义类，继承基础类
    public function insert() {                                //定义方法
        $ins = new Model('user');          //实例化模型类，传递参数为没有前缀的数据表名称
        $ins->Create();                                       //创建数据对象
        $result = $ins->add();                                //写入数据库
        $this->redirect('Index/index','', 5,'页面跳转中');     //页面重定向
    }
}
```

短短的 3 行代码，完成数据的添加操作。其具体应用可以参考例 16.4。

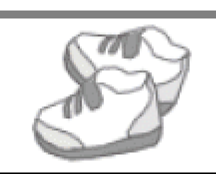
其中的 Create 方法还支持其他方式提交的数据对象。例如，以数组形式提交数据，从其他的数据对象中获取的数据等。其关键代码如下：

```
//数组形式提交数据
$data['user'] = 'mrsoft';
$data['address'] = '长春市';
$User->create($data);
//从 User 数据对象创建新的 Member 数据对象
$User = M("User");                  //实例化 User 对象
$User->find(1);                     //读取数据
$Member = M("Member");              //创建 Member 对象
$Member->create($User);
```

create 方法在创建数据对象的同时，还实现了一些非常有意义的功能，包括支持多种数据源、数据自动验证、字段类型检查和数据自动完成等。

create 方法创建的数据对象保存在内存中，并没有实际写入到数据库中，直到使用 add 或 save 方法才真正将数据添加到数据库中。如果只是想简单创建一个数据对象，那么可以使用 data 方法。例如，实例化 User 模型，通过 data 和 add 方法将数据添加到数据库中，其关键代码如下：

```
$User = M('User');                  //实例化 User 模型
//创建数据后写入到数据库
$data['user'] = 'mrsoft';
$data['address'] = '长春市';
$User->data($data)->add();          //执行数据对象的创建
```





Note

指点迷津：

使用 data 方法创建的数据对象不会进行自动验证和过滤操作，需要自行处理。但在进行 add 或 save 操作时，数据表中不存在的字段以及非法的数据类型（如对象、数组等非标量数据）是会自动过滤的，不用担心因非数据表字段的写入导致 SQL 错误的问题。

16.5.6　连贯操作

ThinkPHP 2.0 版本全面启用模型类的连贯操作方法，可以有效地提高数据存取的代码清晰度和开发效率。例如，查询一个 User 表的满足状态为 1 的前 5 条记录，并按照用户的 ID 排序，其关键代码如下：

```
$User->where('status=1')->order('id')->limit(5)->select();
```

在连贯操作中，select 方法必须放到最后一个，其他的连贯操作方法调用顺序没有先后。如果不习惯使用连贯操作，那么新版本还支持直接使用参数进行查询的方式。例如上面的代码可以改写为：

```
$User->select(array('order'=>'id', 'where'=>'status=1', 'limit'=>'5'));
```

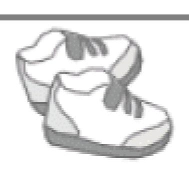
若使用数组参数方式，则索引的名称就是连贯操作的方法名称。其实不仅仅是查询方法可以使用连贯操作，包括 add、save、delete 等方法都可以使用，例如：

```
$User->where('id=1')->field('id,user,address')->find();
$User->where('status=1 and id=1')->delete();
```

下面对连贯操作的方法进行一下总结（更多的用法将在 CURD 操作的过程中详细描述），如表 16.3 所示。

表 16.3　连贯操作方法总结

方　法　名	描　　述
where	用于查询或更新条件的定义。参数支持字符串、数组和对象
table	定义要操作的数据表名称。可以动态改变当前操作的数据表名称，需要写数据表的全名，包含前缀，可以使用别名，例如$Model->table('think_user user')->where('status>1')->select(); table 方法的参数支持字符串和数组，数组方式的用法为： $Model->table(array('think_user'=>'user','think_group'=>'group'))->where('status>1')->select(); 使用数组方式定义的优势是可以避免因为表名和关键字冲突而出错的情况。如果不定义 table 方法，默认会自动获取当前模型对应或者定义的数据表
data	数据对象赋值。可以用于新增或保存数据之前的数据对象赋值，例如 $Model->data($data)->add(); $Model->data($data)->where('id=3')->save(); data 方法的参数支持对象和数组，如果是对象，会自动转换成数组。如果不定义 data 方法赋值，也可以使用 create 方法或者手动给数据对象赋值的方式
field	定义要查询的字段。参数支持字符串和数组，例如： $Model->field('id,nickname as name')->select(); $Model->field(array('id','nickname'=>'name'))->select(); 如果不使用 field 方法指定字段，则默认和使用 field('*')等效
order	对结果进行排序，例如 order('id desc') 排序方法支持对多个字段的排序，例如 order('status desc,id asc') order 方法的参数支持字符串和数组，数组的用法如下： order(array('status'=>'desc','id'))





Note

续表

方　法　名	描　　述
limit	结果限制。在 ThinkPHP 中，无论操作的是 MySQL、MS SQL Server 还是 Oracle 数据库，其 limit 方法是统一的，即 limit('offset,length')，例如 limit('1,10')。获取从第一条记录开始的 10 条记录。注意，limit('10')与 limit('0,10')是等效的
page	查询分页。属于新增特性，可以更加快速地进行分页查询。page()方法的用法和 limit 方法类似，格式为： page('page[,listRows]') page 表示当前的页数，listRows 表示每页显示的记录数。例如 page('2,10')，表示每页显示 10 条记录，获取第 2 页的数据 如果省略 listRow，则会读取 limit('length') 的值，例如 limit(25)->page(3);。表示每页显示 25 条记录，获取第 3 页的数据。如果 limit 也没有设置，则默认为每页显示 20 条记录
group	查询 Group 支持。例如 group('user_id')。group 方法的参数只支持字符串

指点迷津：

有关上述方法的应用，将在后面的 CURD 操作中体现，这里不再一一举例。

16.5.7　CURD 操作

ThinkPHP 提供了灵活和方便的数据操作方法，CURD（创建、更新、读取和删除）是 4 个最基本的数据库操作。CURD 操作通常与连贯操作配合使用。下面将对各种操作的使用方法进行分析（在执行类的实例化操作时，统一使用 M 方法）。

1. 创建操作

在 ThinkPHP 中使用 add 方法完成数据的添加操作。其使用方法如下：

```
$User = M("User"); //  实例化 User 对象
$data['name'] = 'ThinkPHP';
$data['email'] = 'ThinkPHP@gmail.com';
$User->add($data);
```

或者使用 data 方法进行连贯操作。其代码如下：

```
$User->data($data)->add();
```

如果在 add 方法之前已经创建数据对象（例如使用了 create 或 data 方法），则 add 方法就不需要再传入数据了。

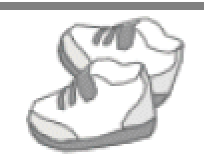
2. 读取数据

在 ThinkPHP 中读取数据的方式很多，通常分为读取某个字段的值、读取数据和读取数据集。读取字段的值使用 getField 方法，读取数据使用 find 方法，读取数据集使用 select 方法。

使用 getField 方法读取某个字段的值，如果传入多个字段，则可以返回一个关联数组。返回的 list 是一个数组，键名是用户的 id，键值是用户的昵称 nickname。例如，获取 ID 为 3 的用户的昵称，获取所有用户的 ID 和昵称列表，代码如下：

```
$User = M("User");                                          //实例化 User 对象
$nickname = $User->where('id=3')->getField('nickname');  //获取 ID 为 3 的用户的昵称
$list = $User->getField('id,nickname');                  //获取所有用户的 ID 和昵称列表
```

select 方法的返回值是一个二维数组，如果没有查询到任何结果，也是返回一个空的数组。

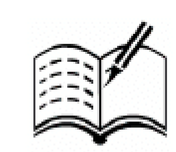


配合上面提到的连贯操作方法，可以完成复杂的数据查询。例如，查找 status 值为 1 的用户数据以创建时间排序返回 10 条数据，代码如下：

```
$User = M("User");                                    //实例化 User 对象
$list = $User->where('status=1')->order('create_time')->limit(10)->select();
```

find 方法与 select 类似，select 可用的所有连贯操作方法也都可以用于 find 方法，区别在于 find 方法最多只会返回一条记录，因此，limit 方法对于 find 查询操作是无效的。例如，查找 status 值为 1.name 值为 think 的用户数据，代码如下：

```
$User = M("User");                                    //实例化 User 对象
$User->where('status=1 and name="think" ')->find();
```



脚下留神：

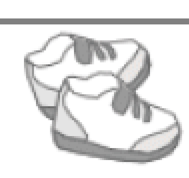
即使满足条件的数据不止一条，find()方法也只会返回第一条记录。

例 16.8 通过 Add 方法向数据库中添加数据，然后查询数据表中用户名等于 mr 的记录，并按照降幂排列，循环输出 3 条记录，其关键操作步骤如下。（**实例位置：配套资源\mr\16\example\16.8**）

这里在讲解示例的实现步骤过程中，省略了项目目录的创建、入口文件的编写和配置文件的设置，其具体步骤可以参考例 16.4 或例 16.5。这里将直接讲解在控制器中如何完成数据的添加和查询操作。

（1）定位到例 16.8\App\Lib\Action\目录下，编写项目控制器。创建 Index 模块，继承系统的 Action 基础类，定义 index 方法，通过 M 方法实例化模型类，应用连贯操作中的 where、order、limit 和 select 方法读取 think_user 数据表中的数据，并且将查询结果赋给模板变量，指定模板页；定义 insert 方法，通过 M 方法实例化模型类，应用 add 方法向指定的数据表中添加数据。IndexAction.class.php 的代码如下：

```
<?php
header("Content-Type:text/html; charset=utf-8");                //设置页面编码格式
class IndexAction extends Action{
    public function index(){
        $db = M('User');                        //实例化模型类，参数数据表名称，不包含前缀
        $select = $db->where('user="mr"')->order('id desc')->limit(3)->select();  //执行查询语句
        $this->assign('select',$select);                        //模板变量赋值
        $this->display();                                       //指定模板页
    }
    public function insert(){
        $dba = M('User');                       //实例化模型类，参数数据表名称，不包含前缀
        $data['user'] = 'mr';
        $data['pass'] = md5('mrsoft');
        $data['address'] = '长春市';
        $result=$dba->add($data);                               //执行添加数据
        if($result){
            $this->redirect('Index/index','', 2,'页面跳转中');  //页面重定向
        }
    }
}
?>
```



Note

（2）定位到 App\Tpl\default 目录下，创建 Index 模块文件夹。编辑 index 操作的模板文件 index.html，应用 ThinkPHP 内置模板引擎中的 foreach 标签循环输出模板变量传递的数据；创建添加数据的表单，将数据提交到控制器的 insert 方法中进行处理。其关键代码如下：

```
<foreach name='select' item='user' >
  <tr class="content">
    <td bgcolor="#FFFFFF"> {$user.id}</td>
    <td bgcolor="#FFFFFF"> {$user.user}</td>
    <td bgcolor="#FFFFFF"> {$user.address}</td>
  </tr>
  </foreach>
<form id="form1" name="form1" method="post" action="__URL__/insert">
<input type="submit" name="button" id="button" value="数据添加" />
</form>
```

运行效果如图 16.16 所示。

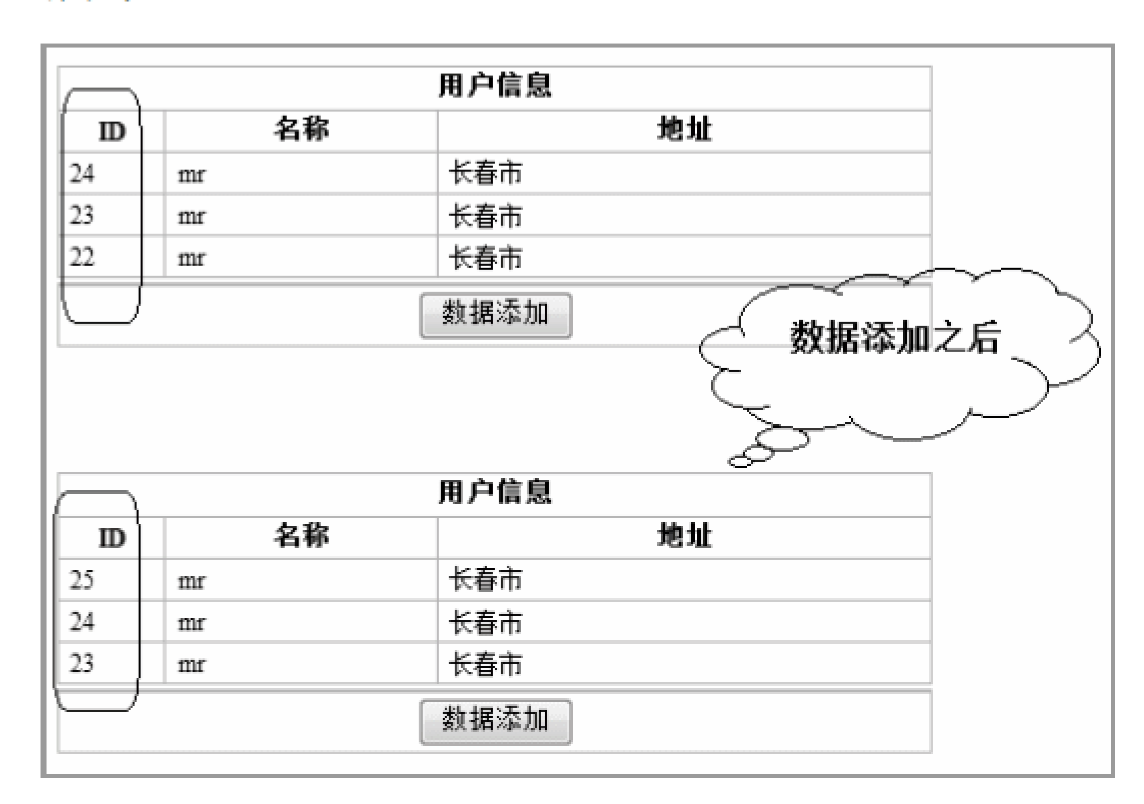


图 16.16　数据添加、查询后的运行结果

3. 更新数据

在 ThinkPHP 中，使用 save 方法更新数据库，并且也支持连贯操作的使用。例如，更新数据表中 name 和 email 字段的值。其代码如下：

```
$User = M("User");                          //实例化 User 对象
$data['name'] = 'ThinkPHP';                 //要修改的数据对象属性赋值
$data['email'] = 'ThinkPHP@gmail.com';
$User->where('id=5')->save($data);          //根据条件保存修改的数据
```

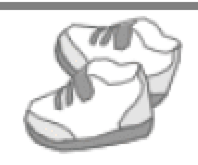
save 方法在执行更新数据的操作时，如果没有设置任何更新条件，且数据对象本身也不包含主键字段，那么 save 方法不会更新任何数据库的记录。

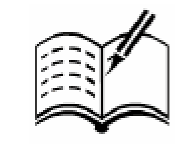
第二种方法是通过 data 方法创建要更新的数据对象，然后通过 save 方法进行保存。例如：

```
$User = M("User");                          //实例化 User 对象
$data['name'] = 'ThinkPHP';                 //要修改的数据对象属性赋值
$data['email'] = 'ThinkPHP@gmail.com';      //要修改的数据对象属性赋值
$User->where('id=5')->data($data)->save();  //根据条件保存修改的数据
```

第三种方法是针对某个字段的值，应用 setField 方法进行更新。例如，更新数据表中字段 name 的值，条件是 ID 为 5 的记录。其代码如下：

```
$User = M("User");                                      //实例化 User 对象
$User-> where('id=5')->setField('name','ThinkPHP');     //更改用户的 name 值
```





如果要更新多个字段的值，也可以应用 setField 方法，只需要传入数组即可，例如：

```
$User = M("User");                                    //实例化 User 对象
//更改用户的 name 和 email 的值
$User-> where('id=5')->setField(array('name','email'),array('ThinkPHP','ThinkPHP@gmail.com'));
```

第四种方法是应用 setInc 和 setDec 方法对统计字段（通常指的是数字类型）中的值进行增减操作。例如，对指定用户的积分进行增、减操作，代码如下：

```
$User = M("User");                                    //实例化 User 对象
$User->setInc('score','id=5',3);                      //用户的积分加 3
$User->setInc('score','id=5');                        //用户的积分加 1
$User->setDec('score','id=5',5);                      //用户的积分减 5
$User->setDec('score','id=5');                        //用户的积分减 1
```

4. 删除数据

在 ThinkPHP 中，使用 delete 方法删除数据库中的记录，同样可以使用连贯操作进行删除操作。例如，删除数据表中 ID 为 5 的记录，代码如下：

```
$User = M("User");                                    //实例化 User 对象
$User->where('id=5')->delete();                       //删除 id 为 5 的用户数据
```

delete 方法可以用于删除单个或多个数据，主要取决于删除条件，也就是 where 方法的参数，也可以用 order 和 limit 方法来限制要删除的个数，例如，删除所有状态为 0 的 5 个用户数据按照创建时间排序，代码如下：

```
$User = M("User");                                    //实例化 User 对象
$User->where('status=0')->order('create_time')->limit('5')->delete();
```

上机演练

上机演练 1　用户信息的查询、更新和删除

应用 ThinkPHP 中的 CURD 操作，实现对用户信息的查询、更新和删除操作。其运行效果如图 16.17 所示。

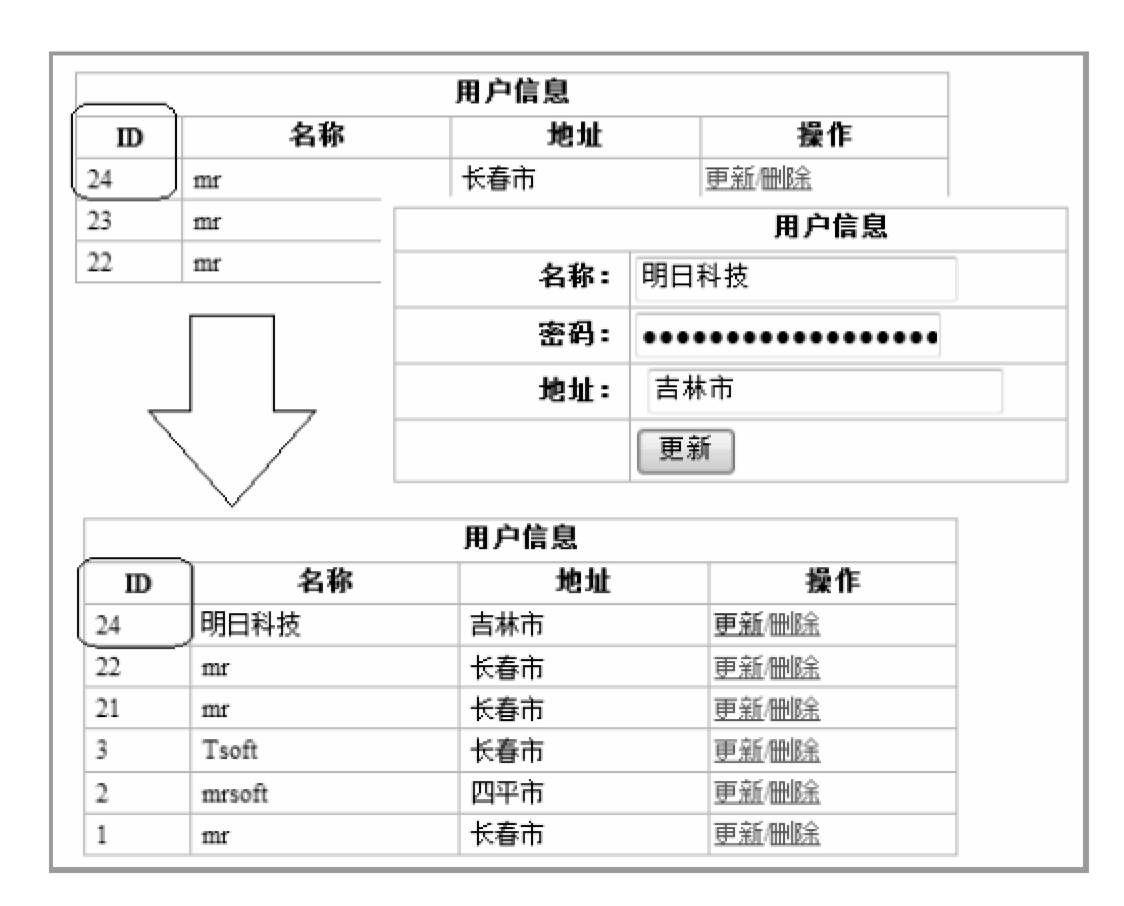
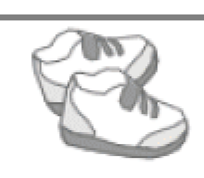


图 16.17　数据更新和删除

这里直接讲解在控制器中如何完成定义 index 方法循环输出数据库中的数据；定义 update 方法完成数据的更新；定义 delete 方法实现数据的删除。





（1）定位到 App\Lib\Action\目录下，编写项目控制器。创建 Index 模块，继承系统的 Action 基础类，定义 index 方法，以记录的 ID 值为条件，降幂循环输出 10 条记录。其代码如下：

```
<?php
header("Content-Type:text/html; charset=utf-8");                    //设置页面编码格式
class IndexAction extends Action{
    public function index(){
        $db = M('User');                         //实例化模型类，参数数据表名称，不包含前缀
        $select = $db->order('id desc')->limit(10)->select();
        $this->assign('select',$select);                    //模板变量赋值
        $this->display();                                   //指定模板页
    }
```

定义 update 方法，首先根据超链接传递 ID 值执行查询，查询出指定的数据，并且将查询结果赋给指定的模板变量。然后判断表单提交的 ID 值是否存在，如果存在则以 ID 为条件，对指定的数据进行更新操作。其关键代码如下：

```
public function update(){
    $db = M('User');                             //实例化模型类，参数数据表名称，不包含前缀
    $select = $db->where('id='.$_GET['id'])->select();
    $this->assign('select',$select);                    //模板变量赋值
    $this->display(update);                             //指定模板页
    if(isset($_POST['id'])){
        $data['user'] = $_POST['user'];                 //要修改的数据对象属性赋值
        $data['pass'] = md5($_POST['pass']);
        $data['address'] = $_POST['address'];
        $result=$db->where('id='.$_POST['id'])->save($data);  //根据条件保存修改的数据
        if($result){
            $this->redirect('Index/index','', 2,'数据更新成功');//页面重定向
        }
    }

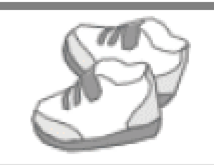
}
```

定义 delete 方法，根据超链接传递的 ID 值，删除数据库中指定的记录。其关键代码如下：

```
 public function delete(){
     $db = M('User');                            //实例化模型类，参数数据表名称，不包含前缀
     $result=$db->where('id='.$_GET['id'])->delete();          //删除 id 为 5 的用户数据
     if($result){
         $this->redirect('Index/index','', 2,'数据删除成功');        //页面重定向
     }
 }
}
?>
```

（2）定位到 App\Tpl\default 目录下，创建 Index 模块文件夹。编辑 index 操作的模板文件 index.html，应用 ThinkPHP 内置模板引擎中的 foreach 标签循环输出模板变量传递的数据；创建更新和删除超链接，将指定记录的 ID 作为参数进行传递。其关键代码如下：

```
<foreach name='select' item='user' >
  <tr class="content">
```



```
<td bgcolor="#FFFFFF"> {$user.id}</td>
<td bgcolor="#FFFFFF"> {$user.user}</td>
<td bgcolor="#FFFFFF"> {$user.address}</td>
<td bgcolor="#FFFFFF"><a href="__URL__/update?id={$user.id}">更新</a>/<a href="__URL__/delete?id={$user.id}">删除</a></td>
</tr>
</foreach>
```

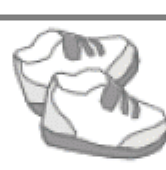
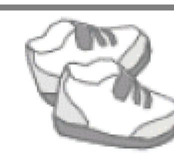
（3）在 Index 模块文件夹下，编辑 update.html 模板文件，创建表单，将从模板变量中读取的数据作为表单元素的默认值进行输出，同时将表单中数据提交到控制器的 update 方法中完成数据的更新操作。其关键代码如下：

```
<form id="form2" name="form2" method="post" action="__URL__/update">
<table width="405" border="1" cellpadding="1" cellspacing="1" bgcolor="#99CC33" bordercolor="#FFFFFF">
<foreach name='select' item='user' >
  <tr class="content">
    <td bgcolor="#FFFFFF" class="right" width="103">名称：</td>
    <td bgcolor="#FFFFFF" width="289"> <input type="hidden" name="id" id="hiddenField" value="{$user.id}" /><input name="user" type="text" id="user" size="20" value="{$user.user}" /></td>
    </tr>
  <tr class="content">
    <td bgcolor="#FFFFFF" class="right">密码：</td>
    <td bgcolor="#FFFFFF"><input name="pass" type="password" id="pass" size="20" value="{$user.pass}" />
     </td>
    </tr>
  <tr class="content">
    <td bgcolor="#FFFFFF" class="right"> 地址：</td>
    <td bgcolor="#FFFFFF"> 
      <input name="address" type="text" id="address" size="30" value="{$user.address}" />
    </td>
    </tr>
  <tr class="content">
    <td bgcolor="#FFFFFF"><input type="submit" name="button" id="button" value="更新" /></td>
  </tr>
  </foreach>
</table>
</form>
```

16.6 ThinkPHP 的视图

视频讲解：配套资源\mr\16\video\ThinkPHP 的视图.exe

在 ThinkPHP 中，视图由两部分组成：View 类和模板文件。Action 控制器直接与 View 视

图类进行交互，把要输出的数据通过模板变量赋值的方式传递到视图类，而具体的输出工作则交由 View 视图类来进行，同时视图类还完成了一些辅助工作，包括调用模板引擎、布局渲染、输出替换、页面 Trace 等功能。为了方便使用，在 Action 类中封装了 View 类的一些输出方法，如 display、fetch、assign、trace 和 buildHtml 等方法，这些方法的原型都在 View 视图类中。



Note

16.6.1 模板定义

为了实现对模板文件更加有效的管理，ThinkPHP 对模板文件进行了目录划分，默认的模板文件定义规则是：

```
模板目录/模板主题/[分组名/]模块名/操作名+模板后缀
```

模板目录默认是项目下面的 Tpl，模板主题默认是 default，模板主题功能是为了多模板切换而设计的。如果有多个模板主题，则可以使用 DEFAULT_THEME 参数设置默认的模板主题名。

在每个模板主题下面是以项目的模块名为目录，然后是每个模块的具体操作模板文件，例如，User 模块的 add 操作对应的模板文件是 Tpl/default/User/add.html。

模板文件的默认后缀是.html，后缀可以通过 TMPL_TEMPLATE_SUFFIX 来配置。

如果项目启用模块分组功能（假设 User 模块属于 Home 分组），那么默认对应的模板文件就会发生变化，变为 Tpl/default/Home/User/add.html。

分组功能可以通过 TMPL_FILE_DEPR 参数来配置，进而简化模板的目录层次。例如，设置 TMPL_FILE_DEPR 等于“_”，那么默认的模板文件就变成 Tpl/default/Home/User_add.html。

指点迷津：

正是因为系统有了这样一种模板文件自动识别的规则，display 方法才可以无需带任何参数就输出对应的模板。

16.6.2 模板赋值

模板赋值是在 Action 控制器中完成的，通过 assign 方法将控制器中获取的数据赋给模板变量。例如：

```
$this->assign('name',$value);
```

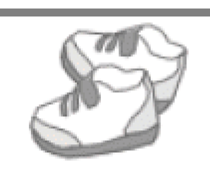
如果要同时输出多个模板变量，则可以使用数组的方式进行赋值：

```
$array = array();
$array['name']= 'thinkphp';
$array['email']='liu21st@gmail.com';
$array['phone']='12335678';
$this->assign($array);
```

这样，就可以在模板文件中同时输出 name、email 和 phone 3 个变量。

16.6.3 指定模板文件

模板变量赋值后就需要调用模板文件来输出相关的变量，模板调用应用的是 display 方法。





下面讲解如何通过 display 方法完成对模板的调用，如表 16.4 所示。

表 16.4　display 方法的应用

语 法 格 式	描　　述
display('操作名')	调用当前模块的其他操作模板。例如，当前是 User 模块下面的 read 操作，而要调用 User 模块的 edit 操作模版，使用$this->display('edit');
display('分组名:模块名:操作名')	调用其他模块的操作模板。其中分组名是可选的。例如，当前是 User 模块，要调用 Member 模块的 read 操作模版，使用$this->display('Member:read'); 如果要调用分组 Admin 的 Member 模块的 read 操作模板，使用$this->display('Admin:Member:read');
display('主题名@模块名:操作名')	调用其他主题的操作模板。例如，调用 Admin 主题的 User 模块的 edit 操作模版，使用$this->display('Admin@User:edit'); 。此种方式需要指定模块和操作名
display('模板文件名')	直接全路径输出模板。例如，直接输出当前的 Public 目录下面的 menu.html 模板文件，使用$this->display('./Public/menu.html');这种方式需要指定模板路径和后缀，这里的 Public 目录位于当前项目入口文件位置下面。如果是其他的后缀文件，也支持直接输出，例如$this->display ('./Public/menu.tpl');，只要./Public/menu.tpl 是一个实际存在的模板文件。如果使用的是相对路径，要注意当前位置是相对于项目的入口文件，而不是模板目录
display('模板文件名', 'charset')	设置模板页的编码。例如，设置指定模板页的编码为 gbk，使用$this->display('Member:read', 'gbk');
display('模板文件名', 'charset', 'format')	设置指定模板文件的编码和格式。例如，设置模板文件为 UTF-8 编码、文件为 XML 格式。其应用为$this->display('Member:read', 'utf-8', 'text/xml');

指点迷津：

在第二种用法中，不需要写模板文件的路径和后缀，严格来说，这里面的模块名和操作名并不一定需要有对应的模块或操作，只是一个目录名称和文件名称而已，例如，项目中可能没有 Public 模块，更没有 Public 模块的 menu 操作，但是一样可以使用$this->display('Public:menu');输出该模板文件。

模板变量赋值后，在指定的模板文件中进行输出，具体的输出方法需要根据选择的模板引擎来决定。如果使用的是内置的模板引擎，可参考 ThinkPHP 开发完全手册模板指南中的内容；如果使用 PHP 本身作为模板引擎，则直接在模板文件中输出，例如<?php echo $name.'['.$email.''.$phone.']';?>。

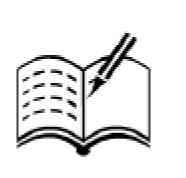
16.6.4　特殊字符串替换

在进行模板输出之前，系统还会对模板的特殊字符串进行替换，实现模板输出的替换和过滤。该机制使得模板文件的定义更加方便，默认的替换规则如表 16.5 所示。

表 16.5　模板中特殊字符串的替换规则

特殊字符串	替 换 描 述
../Public	被替换成当前项目的公共模板目录。通常是：/项目目录/Tpl/default/Public/
__PUBLIC__	被替换成当前网站的公共目录。通常是：/Public/
__TMPL__	替换成项目的模板目录。通常是：/项目目录/Tpl/default/





Note

续表

特殊字符串	替 换 描 述
__ROOT__	会替换成当前网站的地址（不含域名）
__APP__	替换成当前项目的 URL 地址（不含域名）
__URL__	替换成当前模块的 URL 地址（不含域名）
__ACTION__	替换成当前操作的 URL 地址（不含域名）
__SELF__	替换成当前的页面 URL

脚下留神：

特殊字符串是严格区分大小写的，并且这些特殊字符串的替换规则是可以更改或增加的。只要在项目配置文件中配置 TMPL_PARSE_STRING 就可以完成。如果有相同的数组索引，就会更改系统的默认规则。例如：

```
TMPL_PARSE_STRING    => array(
        '__PUBLIC__' => '/Common',           //更改默认的__PUBLIC__ 替换规则
        '__UPLOAD__' => '/Public/Uploads/',   //增加新的上传路径替换规则
)
```

上机演练

上机演练 2　用户登录和数据的分页输出

实现用户登录功能，将登录用户的信息存储到 SESSION 变量中，应用 ThinkPHP 中提供的分页扩展类和 Page 方法完成数据的分页输出。其运行效果如图 16.18 所示。



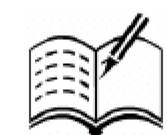
图 16.18　用户登录和数据的分页输出

（1）定位到\App\Lib\Action\目录下，编写项目控制器。创建 Index 模块，继承系统的 Action 基础类，定义 index 方法，验证用户提交的用户名和密码是否正确，如果正确则将登录用户名存储到 SESSION 变量中，并且将网页重定向到 main.html 页面。其代码如下：

```
<?php
session_start();                                        //初始化 SESSION 变量
header("Content-Type:text/html; charset=utf-8");        //设置页面编码格式
class IndexAction extends Action{
    public function index(){
        if(isset($_POST['user'])){
            if(isset($_POST['user']) && isset($_POST['pass'])){
                $db = M();                  //实例化模型类，参数数据表名称，不包含前缀
                $select = $db->query("select * from think_user where user='".$_POST['user']."'
and pass='".$_POST['pass']."'");            //执行查询语句，验证用户名和密码是否正确
                if($select){
                    $_SESSION['admin']=$_POST['user'];   //将登录用户名存储到 SESSION 中
                        $this->redirect('Index/main','', 2,'用户 '.$_POST['user'].' 登录成功！');
                                            //页面重定向
```



```
                }else{
                    $this->redirect('Index/index','', 2,'用户名或者密码不正确！'); //页面重定向
                }
            }else{
                $this->redirect('Index/index','', 2,'用户名、密码不能为空！');      //页面重定向
            }
        }
        $this->display();
    }
```



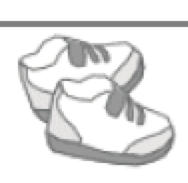
定义 main 方法，载入分页类，完成数据库中数据的分页查询，并且将查询结果赋给模板变量。其代码如下：

```
public function main(){
    $db = M('User');                    //实例化模型类，参数数据表名称，不包含前缀
    //进行分页数据查询，注意 page()方法的参数的前面部分是当前的页数，使用$_GET[p]获取
    if(isset($_GET['p'])){              //判断分页变量是否存在
        $p=$_GET['p'];
    }else{
        $p=1;
    }
    $list = $db->where('address='.'"长春市"')->order('id desc')->page($p.',1')->select(); //查询数据
    $this->assign('select',$list);      //赋值数据集
    import("ORG.Util.Page");            //导入分页类
    $count = $db->where('address='.'"长春市"')->count();     //查询满足要求的总记录数
    $Page = new Page($count,1);         //实例化分页类，传入总记录数和每页显示的记录数
    $show = $Page->show();              //分页显示输出
    $this->assign('page',$show);        //赋值分页输出
    $this->display(main);               //输出模板
}
```

定义 validatorcode 方法，应用 GD 库中的函数，根据超链接传递的值生成用户登录的验证码。其代码如下：

```
public function validatorcode(){
    header('content-type:image/png');                                   //定义标题 PNG 格式图像
    $im = imagecreate(65, 25);                                          //定义画布
    imagefill($im, 0, 0, imagecolorallocate($im, 200, 200, 200));       //区域填充
    $validatorCode = $_GET['code'];                                     //获取提交的值
    imagestring($im, rand(3, 5), 10, 3, substr($validatorCode, 0, 1), imagecolorallocate($im, 0,
rand(0, 255), rand(0, 255)));
    imagestring($im, rand(3, 5), 25, 6, substr($validatorCode, 1, 1), imagecolorallocate($im,
rand(0, 255), 0, rand(0, 255)));
    imagestring($im, rand(3, 5), 36, 9, substr($validatorCode, 2, 1), imagecolorallocate($im,
rand(0, 255), rand(0, 255), 0));
    imagestring($im, rand(3, 5), 48, 12, substr($validatorCode, 3, 1), imagecolorallocate($im, 0,
rand(0, 255), rand(0, 255)));
    imagepng($im);                                                      //生成 PNG 图像
    imagedestroy();                                                     //销毁图像
}
```

（2）定位到 App\Tpl\default 目录下，创建 Index 模块文件夹。编辑 index 操作的模板文件



Note

index.html，载入 CSS 样式文件和 JavaScript 文件，创建表单，完成用户登录信息的提交操作。其关键代码如下：

```
<link href="__ROOT__/Public/Css/style.css" rel="stylesheet" type="text/css" />
<js href="__ROOT__/Public/Js/check.js" />
<form name="form1" method="post"   action="__URL__/index" onSubmit="return chkinput(this)" >
<table width="265" border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td class="title" id="td">用户名：</td>
        <td><input name="user" type="text" size="15" /></td>
    </tr>
    <tr>
        <td class="title" id="td">密码：</td>
        <td><input name="pass" type="password" size="15" /></td>
    </tr>
    <tr>
        <td class="title" id="td">验证码：</td>
        <td>
            <input type="text" name="validatorCode" size="10" />
            <input type="hidden" name="defValidatorCode" value="" />
<script language="javascript">
    var num1=Math.round(Math.random()*10000000);          //生成随机数
        var num=num1.toString().substr(0,4);               //截取随机数的前 4 个字符
        document.write("<img name=codeimg src='__URL__/validatorcode?code="+num+"'>");
                                                            //将截取值传递到图像处理页中
        form1.defValidatorCode.value=num;                   //将截取值赋给表单中的隐藏域
    function reCode(){                                      //定义方法，重新生成验证码
        var num1=Math.round(Math.random()*10000000);        //生成随机数
        var num=num1.toString().substr(0,4);                //截取随机数
        document.codeimg.src="__URL__/validatorcode?code="+num;//将截取值传递到图像处理
页中
        form1.defValidatorCode.value=num;                   //将截取值赋给表单中的隐藏域
    }
</script>
            <a href="javascript:reCode()" class="content">看不清</a>
        </td>
    </tr>
</table>
    <input type="image" name="imageField" id="imageField" src="__ROOT__/Public/images/66_05.gif" />
</form>
```

（3）在 Index 模块文件夹下编辑 main.html 文件，通过模板引擎中的 session 标签输出当前登录的用户名，通过 foreach 标签循环输出模板变量传递的数据，最后输出模板变量传递的分页超链接。其关键代码如下：

```
<table width="405" border="1" cellpadding="1" cellspacing="1" bgcolor="#99CC33" bordercolor=
"#FFFFFF">
    <tr>
    <td colspan="3" bgcolor="#FFFFFF" class="title" align="center">当前登录用户：{$Think.session.
```



```
admin}</td>
                </tr>
                <foreach name='select' item='user' >
                <tr class="content">
                  <td bgcolor="#FFFFFF"> {$user.id}</td>
                  <td bgcolor="#FFFFFF"> {$user.user}</td>
                  <td bgcolor="#FFFFFF"> {$user.address}</td>
                </tr>
                </foreach>
                 <tr class="content">
                  <td colspan="3" bgcolor="#FFFFFF"> {$page}</td>
                  </tr>
              </table>
```

16.7 内置 ThinkTemplate 模板引擎

视频讲解：配套资源\mr\16\video\内置 ThinkTemplate 模板引擎.exe

ThinkPHP 内置了一个基于 XML 的性能卓越的模板引擎 ThinkTemplate，这是一个专门为 ThinkPHP 服务的内置模板引擎。它使用 XML 标签库技术编制，支持两种类型的模板标签（普通标签和 XML 标签），使用了动态编译和缓存技术，而且支持自定义标签库。

ThinkTemplate 模板引擎生成的编译文件默认存储于 Runtime/Cache 目录下，以模板文件的 md5 编码作为缓存文件名保存。

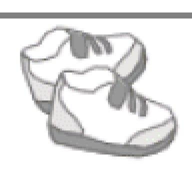
下面介绍一些 ThinkTemplate 模板引擎中的常用标签，如表 16.6 所示。

表 16.6　ThinkTemplate 模板引擎中的常用标签

标 签 名 称	应 用 描 述
{$name}	输出模板引擎中的变量。注意模板标签的“{”和“$”之间不能有空格，否则标签无效

//输出$_SERVER 变量
{$Think.server.script_name }
//输出$_SESSION 变量
{$Think.session.session_id|md5 }
//输出$_GET 变量
{$Think.get.pageNumber }
//输出$_COOKIE 变量
{$Think.cookie.name }
{$Think.now }　　　　//现在时间
{$Think.template|basename }//模板页面
	系统变量。除了常规变量的输出外，模板引擎还支持系统变量和系统常量以及系统特殊变量的输出。它们的输出不需要事先赋值给某个模板变量。系统变量的输出必须以$Think.开头，并且仍然支持使用函数

{$变量|default="默认值"}	默认值输出。如果输出的模板变量没有值，但是需要在显示时赋予一个默认值，可以使用 default 语法。例如： {$user.nickname\|default="明日科技"} 对系统变量的输出也可以支持默认值，例如： {$Think.post.name|default="名称为空"}



续表

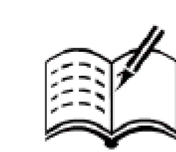
标 签 名 称	应 用 描 述

//使用完整文件名包含
<include file="完整模板文件名" />
//包含当前模块的其他操作模板文件
<include file="操作名" />
//包含其他模块的操作模板
<include file="模块名:操作名" />
//包含其他模板主题的模块操作模板
<include file="主题名@模块名:操作名" />
//用变量控制要导入的模板
<include file="$变量名" />
	使用 Include 标签来包含外部的模板文件
	① 完整文件名的包含。例如：<include file="./Tpl/default/Public/header.html" />。这种情况下，模板文件名必须包含后缀。使用完整文件名包含时，特别要注意文件包含指的是服务器端包含，而不是包含一个 URL 地址，也就是说，file 参数的写法是服务器端的路径，如果使用相对路径，则是基于项目的入口文件位置。
	② 用变量包含。例如：<include file="$tplName" />。为 $tplName 赋不同的值就可以包含不同的模板文件，变量的值的用法和上面的用法相同。
	注意：由于模板解析的特点，从入口模板开始解析，如果外部模板有所更改，模板引擎并不会重新编译模板，除非缓存已经过期。如果修改了包含的外部模板文件，需要把模块的缓存目录清空，否则无法生效

<import type='js' file="Js.Util.Array" /> <import type='css' file="Css.common" /> <load href="../Public/Js/Common.js" /> <load href="../Public/Css/common.css" /> <js href="__PUBLIC__/Js/Common.js" /> <css href="../Public/Css/common.css" />	导入文件。系统提供专门 import 和 load 标签完成文件的导入操作。第一个是 import 标签，导入方式采用类似 ThinkPHP 的 import 函数的命名空间方式。import 标签默认的起始路径是网站的 Public 目录，如果需要指定其他的目录，可以使用 basepath 属性，例如： <import file="Js.Util.Array" basepath="./Common" /> 第二个是 load 标签，通过文件方式导入当前项目的公共 JS 或 CSS。在 href 属性中可以使用特殊模板标签替换，例如： <load href="__PUBLIC__/Js/Common.js" /> Load 标签可以无需指定 type 属性，系统会自动根据后缀自动判断 系统还提供了两个标签别名 js 和 css，其用法和 load 一致

<volist name="list" id="vo" offset="5" length='10'>
  {$vo.name}
</volist>
	volist 标签主要用于在模板中循环输出数据集或多维数组
	参数 Name：表示模板赋值的变量名称，不可随意在模板文件中改变
	参数 Id：表示当前的循环变量，可以随意指定，但确保不要和 name 属性冲突，支持输出部分数据，例如输出其中的第 5～15 条记录
	参数 Offset：表示记录的起始位置
	参数 Length：表示记录的长度
	参数 Mod：控制输出记录的奇偶性，还可以控制在指定的记录换行。例如：
	//输出偶数记录
	<volist name="list" id="vo" mod="2" >
	<eq name="mod" value="1">{$vo.name}</eq>
	</volist>
	//控制一定记录的换行
	<volist name="list" id="vo" mod="5" >
	{$vo.name}
	<eq name="mod" value="4">
	</eq>
	</volist>





续表

标 签 名 称	应 用 描 述
<foreach name="list" item="vo" > {$vo.id} {$vo.name} </foreach>	foreach 标签用于循环输出，它比 volist 标签简洁，但比 volist 标签功能较少。优势是可以对对象进行遍历输出，而 volist 标签通常是用于输出数组
<switch name="变量" > <case value="值 1">输出内容 1</case> <case value="值 2">输出内容 2</case> <default　/>默认情况 </switch>	switch 标签，类似于 PHP 中的 switch 语句，其中 name 属性可以使用函数以及系统变量，例如： <switch name="Think.get.userId|abs"> <case value="1">admin</case> <default />default </switch> 对于 case 的 value 属性可以支持多个条件的判断，使用“|”进行分割，例如： <switch name="Think.get.type"> <case value="gif|png|jpg">图像格式</case> <default />其他格式 </switch> 表示如果$_GET["type"] 是 GIF、PNG 或 JPG 格式，就判断为图像格式。 也可以对 case 的 value 属性使用变量，例如： <switch name="User.userId"> <case value="$adminId">admin</case> <case value="$memberId">member</case> <default />default </switch> 使用变量方式的情况下，不再支持多个条件的同时判断

上机演练

上机演练 3　应用 ThinkPHP 中提供的验证码类和分页类完成用户登录和分页输出

仍然以用户登录和数据的输出为背景，应用 ThinkPHP 中提供的验证码类和分页类生成验证码，完成数据的分页输出。其运行效果如图 16.19 所示。

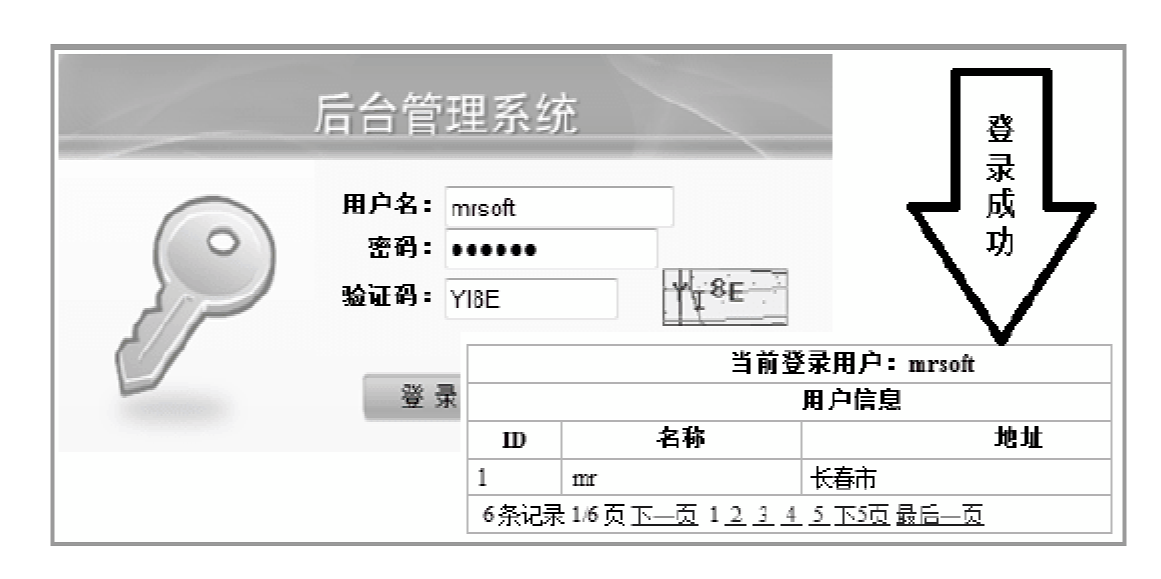
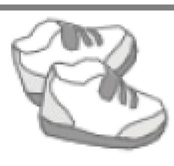
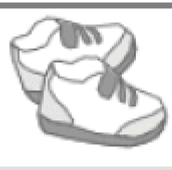


图 16.19　验证码类和分页类的应用效果

（1）定位到\App\Lib\Action\目录下，编写项目控制器。创建 Index 模块，继承系统的 Action 基础类，定义 index 方法，验证 SESSION 变量存储的验证码与用户提交的验证码是否相同，验证用户提交的用户名和密码是否正确，如果正确则将登录用户名存储到 SESSION 变量中，并且将网页重定向到 main.html 页面。其代码如下：

```
<?php
session_start();
```

Note

```
header("Content-Type:text/html; charset=utf-8");          //设置页面编码格式
class IndexAction extends Action{
  public function index(){
        if(isset($_POST['user'])){
              if(isset($_POST['user']) && isset($_POST['pass']) && isset($_POST['validatorCode'])){
                    if($_SESSION['verify'] == md5($_POST['validatorCode'])) {      //验证验证码是否正确
                          $db = M();            //实例化模型类，参数数据表名称，不包含前缀
                          $select = $db->query("select * from think_user where user='".$_POST['user']."' and pass='".$_POST['pass']."'"); //执行查询语句，验证用户名和密码是否正确
                          if($select){
                                $_SESSION['admin']=$_POST['user'];
                                $this->redirect('Index/main','', 2,'用户 '.$_POST['user'].' 登录成功！'); //页面重定向
                          }else{
                                $this->redirect('Index/index','', 2,'用户名或者密码不正确！');//页面重定向
                          }
                    }else{
                          $this->redirect('Index/index','', 2,'验证码不正确！');          //页面重定向
                    }
              }else{
                    $this->redirect('Index/index','', 2,'用户名、密码不能为空！'); //页面重定向
              }
        }
        $this->display();
  }
```

定义 main 方法，载入分页类，完成数据库中数据的分页查询，并且将查询结果赋给模板变量。这里应用的是 Page 类和 limit 方法完成数据的分页输出，其代码如下：

```
public function main(){
      $db = M('User');                  //实例化模型类，参数数据表名称，不包含前缀
      import("ORG.Util.Page");                          //导入分页类
      $count = $db->count();                            //统计总记录数
      //$count = $User->where("status=1")->count();     //查询满足要求的总记录数
      $Page = new Page($count,1);       //实例化分页类，传入总记录数和每页显示的记录数
      $show = $Page->show();                            //分页显示输出
      // 进行分页数据查询，注意 limit()方法的参数要使用 Page 类的属性
      $list = $db->order('id')->limit($Page->firstRow.','.$Page->listRows)->select();
      $this->assign('select',$list);                    //赋值数据集
      $this->assign('page',$show);                      //赋值分页输出
      $this->display(main);                             //输出模板
}
```

定义 verify 方法，载入 ThinkPHP 中提供的验证码扩展类，调用 buildImageVerify 方法生成验证码。其代码如下：

```
public function verify(){
    import("ORG.Util.Image");                           //载入验证码类
```



```
        image::buildImageVerify(4,5,'png','65','30','verify');    //生成验证码
    }
```

指点迷津：

buildImageVerify 方法的语法如下：

buildImageVerify($length,$mode,$type,$width,$height,$verifyName)

参数说明如表 16.7 所示。

表 16.7 验证码类中 buildImageVerify 方法的参数及说明

参　　数	说　　明
length	验证码的长度，默认为 4 位数
mode	验证字符串的类型，默认为数字，其他支持类型有：0 字母；1 数字；2 大写字母；3 小写字母；4 中文；5 混合（去掉了容易混淆的字符 oOLl 和数字 01）
type	验证码的图片类型，默认为 PNG 格式
width	验证码的宽度，默认会自动根据验证码长度自动计算
height	验证码的高度，默认为 22
verifyName	验证码的 SESSION 记录名称，默认为 verify

生成验证码之后，需要在模板页中通过输出生成的验证码图像；在控制器中通过如下代码验证用户提交的验证码是否正确：

```
if($_SESSION['verify'] != md5($_POST['verify'])) {
    $this->error('验证码错误！');
}
```

（2）定位到 App\Tpl\default 目录下，创建 Index 模块文件夹。编辑 index 操作的模板文件 index.html，载入 CSS 样式文件和 JavaScript 文件，创建表单，完成用户登录信息的提交操作，然后通过 img 标签输出生成的验证码。其关键代码如下：

```
<link href="__ROOT__/Public/Css/style.css" rel="stylesheet" type="text/css" />
<js href="__ROOT__/Public/Js/check.js" />
<form name="form1" method="post"  action="__URL__/index" onSubmit="return chkinput(this)" >
<table width="265" border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td class="title" id="td">用户名：</td>
        <td><input name="user" type="text" size="15" /></td>
    </tr>
    <tr>
        <td class="title" id="td">密码：</td>
        <td><input name="pass" type="password" size="15" /></td>
    </tr>
    <tr>
        <td class="title" id="td">验证码：</td>
        <td><input type="text" name="validatorCode" size="10" /></td>
        <td><img src="__APP__/Index/verify/" /></td>
    </tr>
</table>
</form>
```



Note

（3）在 Index 模块文件夹下编辑 main.html 文件，通过模板引擎中的 session 标签输出当前登录的用户名，然后通过 foreach 标签循环输出模板变量传递的数据，最后输出模板变量传递的分页超链接。其关键代码如下：

```
        <table width="405" border="1" cellpadding="1" cellspacing="1" bgcolor="#99CC33" bordercolor=
"#FFFFFF">
          <tr>
            <td colspan="3" bgcolor="#FFFFFF" class="title" align="center">当前登录用户：{$Think.
session.admin}</td>
          </tr>
          <foreach name='select' item='user' >
          <tr class="content">
            <td bgcolor="#FFFFFF"> {$user.id}</td>
            <td bgcolor="#FFFFFF"> {$user.user}</td>
            <td bgcolor="#FFFFFF"> {$user.address}</td>
          </tr>
          </foreach>
           <tr class="content">
            <td colspan="3" bgcolor="#FFFFFF"> {$page}</td>
            </tr>
        </table>
```

脚下留神：

这里应用的验证码是区分字母大小写的。

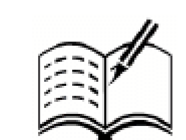
本 章 摘 要

1．了解 ThinkPHP 框架。
2．掌握 ThinkPHP 框架开发项目流程。
3．ThinkPHP 项目结构。
4．ThinkPHP 中的配置方法。
5．ThinkPHP 中控制器的应用。
6．ThinkPHP 中模型的应用，包括模型实例化、属性访问、连接数据库、创建数据、执行连贯操作以及执行数据的添加、更新和删除操作等。
7．ThinkPHP 视图，完成数据的输出操作。
8．ThinkPHP 模板引擎，在视图文件中应用内置模板引擎提供的标签操作数据。

习 题

1．ThinkPHP 框架中目录分为两部分：（　　）和（　　）。
A．系统目录　　B．项目目录　　C．前台目录　　D．后台目录
2．ThinkPHP 框架中的（　　）目录可以自动创建？
A．系统　　B．项目　　C．前台　　D．后台





3．ThinkPHP 框架支持（　　）文件操作？

A．单入口　　B．双入口　　C．多入口　　D．无入口

4．下面有关 ThinkPHP 框架开发项目的规范描述中，哪些是正确的？（　　）

A．函数的命名使用小写字母和下划线的方式，例如 get_client_ip

B．Action 控制器类以 Action 为后缀，例如 UserAction、InfoAction

C．模型类以 Model 为后缀，例如 UserModel、InfoModel

D．方法的命名使用驼峰法，并且首字母小写，例如 getUserName

5．ThinkPHP 提供了灵活的全局配置功能，采用最有效率的 PHP 返回数组方式定义，其支持的配置包括（　　）。

A．惯例配置　　B．项目配置　　C．调试配置　　D．模块配置

6．ThinkPHP 中项目配置的文件默认存储于（　　）下。

7．ThinkPHP 的控制器就是模块类，通常位于项目的（　　）目录下。

8．ThinkPHP 提供了灵活和方便的数据操作方法 CURD，它代表（　　）操作。

9．模板文件的默认后缀是（　　），后缀可以通过（　　）来配置。

10．在 Action 控制器中，通过（　　）将控制器中获取的数据赋给模板变量。

实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 15 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　通过 ThinkPHP 中的扩展类生成中文验证码

关键是载入 ThinkPHP\Lib\ORG\Util\ Image.class.php 中的 Image 类，调用其中的 GBVerify 方法，生成中文验证码。其运行效果如图 16.20 所示。

脚下留神：

在 ThinkPHP 2.0 版本提供的 Image 类中，在定义 GBVerify()方法时有一处错误，需要读者手动修改。其中“$codex = msubstr($code, $i, 1);”对字符串进行截取调用的 msubstr 方法不存在。正确的调用方法是“$codex = String::msubstr($code, $i, 1);”

实战模拟 2　带查询条件的分页

通过 ThinkPHP 框架开发一个站内搜索的功能，并且对查询结果进行分页输出。其运行结果如图 16.21 所示。



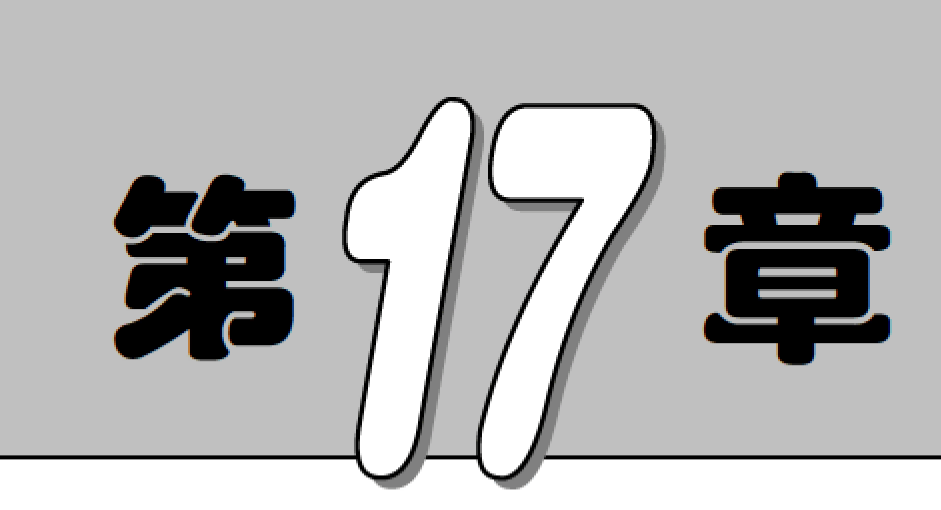
图 16.20　中文验证码

站内搜索：长春　提交

用户信息		
ID	名称	地址
21	mr	长春市

4 条记录 2/4 页 上一页 下一页 1 2 3 4

图 16.21　带查询条件的分页



PHP 的字符编码

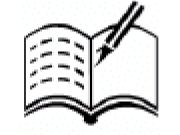
（ 自学视频、源程序：配套资源\mr\17\）

网页字符编码就是计算机在显示字符时使用的编码，也就是一个字符集。在 PHP 中，为网页设置准确、适合的字符编码是非常重要的，只有使用正确的字符编码，网页才能够流畅、清晰地输出数据，否则网页中将会输出乱码。本章将详细地讲解如何在 PHP 中设置字符编码以及出现字符编码问题的解决方法。

学习摘要：

- 字符集和编码
- PHP 网页中设置编码格式
- 编码格式转换
- 检测字符串的编码
- 解决页面中的乱码问题
- 数据库中字符集的编码问题
- 避免截取中文字符串时出现乱码





17.1 字符集和编码

字符是各种文字和符号的总称，包括各国家文字、数字、标点符号和图形符号等。字符集和编码经常出现在一起，但它们实际上一个是名词一个是动词。字符集是将人类使用的自然文字映射到计算机内部二进制的表示方法，是某种文字和字符的集合，作为名词使用，例如 GB2312 字符集、ISO8859 字符集；而编码是对这种字符集的编码方式，作为动词使用。

计算机中的“字符”都是用一个二进制数字来表示的，这个二进制数字称为字符的编码，用于不同国家的语言在计算机中的存储和解释规范，而针对不同需求选择的字符集合，被定义为不同的字符集，例如 GB2312。

中文文字数目大，并且还分为简体中文和繁体中文两种不同书写规则，而计算机最初是按英语单字节字符设计的，因此，对中文字符进行编码是中文信息交流的技术基础。

17.1.1 ISO8859 字符集

ISO8859 是由 ISO（International Organization for Standardization，国际标准化组织）在 ASCII 编码基础上制定的编码标准，该标准包括 128 个 ASCII 字符，并新增加了 128 个字符，用于西欧国家的符号。

ISO8859 存在不同的语言家族：

- ☑ Latin-1（西欧语言）。
- ☑ Latin-2（非 Cyrillic 的中欧和东欧语言）。
- ☑ Latin-3（南欧语言）。
- ☑ Latin-4（土耳其语言）。
- ☑ Latin-5（北欧和波罗地语言）。
- ☑ Latin-6（Cyrillic 西里尔语言）。
- ☑ Latin-7（阿拉伯语言）。
- ☑ Latin-8（希伯来语言）。

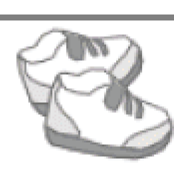
17.1.2 GB2312 与 GBK 字符集

GB2312 又称为 GB2312-80 编码，全称为《信息交换用汉字编码字符集·基本集》，由原中国国家标准总局发布，1981 年 5 月 1 日实施。GB2312 编码是中华人民共和国国家标准汉字信息交换用编码，它所收录的汉字已经覆盖 99.75%的使用频率，基本满足了汉字的计算机处理需要，在中国大陆和新加坡获广泛使用。

GB2312 编码收录简化汉字及一般符号、序号、数字、拉丁字母、日文平假名、希腊字母、俄文字母、汉语拼音符号、汉语注音字母，共 7445 个图形字符。其中包括 6763 个汉字，其中一级汉字 3755 个，二级汉字 3008 个；包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母在内的 682 个全角字符。

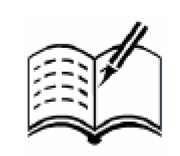
GB2312 编码中对所收汉字进行了“分区”处理，每区含有 94 个汉字/符号。这种表示方式也称为区位码。

各区包含的字符如下：01～09 区为特殊符号；16～55 区为一级汉字，按拼音排序；56～



87 区为二级汉字，按部首/笔画排序；10～15 区及 88～94 区则未有编码。

两个字节中前面的字节为第一字节，后面的字节为第二字节。习惯上称第一字节为“高字节”，而称第二字节为“低字节”。“高位字节”使用了 0xA1～0xF7（把 01～87 区的区号加上 0xA0），“低位字节”使用了 0xA1～0xFE（把 01～94 加上 0xA0）。



以 GB2312 编码的第一个汉字“啊”字为例，它的区号 16，位号 01，则区位码是 1601，在大多数计算机程序中，高字节和低字节分别加 0xA0 得到程序的汉字处理编码 0xB0A1。计算公式是：0xB0=0xA0+16，0xA1=0xA0+1。

GBK 编码是对 GB2312 编码的扩充，它包含两万多个字符，除了保持和 GB2312 兼容外，还包含并且增加了部分 Unicode 中没有的字符。需要注意的是，GBK 只是一个规范而不是国家标准的编码，新的国家标准是 GB18030-2000，它比 GBK 包含更多的字符。

17.1.3　Unicode 字符集

Unicode 字符集是 Universal Multiple-Octet Coded Character Set 通用多八位编码字符集的简称，是由一个名为 Unicode 学术学会（Unicode Consortium）的机构制订的字符编码系统，支持现今世界各种不同语言的书面文本的交换、处理及显示。该编码于 1990 年开始研发，1994 年正式公布，最新版本是 2005 年 3 月 31 日的 Unicode 4.1.0。

Unicode 是一种在计算机上使用的字符编码。它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。

Unicode 标准始终使用十六进制数字，而且在书写时在前面加上前缀“U+”，例如，字母“A”的编码为 004116，字符“□”的编码为 20AC16，所以“A”的编码书写为“U+0041”。

Unicode 堪称是一个伟大的成就，它把在这个世界上的每一个合理的文字系统整合成了一个单一的字符集。

17.1.4　UTF-8 编码

UTF-8 编码是 Unicode 中的一个使用方式。UTF 是 Unicode Translation Format，即把 Unicode 转做某种格式的意思。

UTF-8 便于不同的计算机之间使用网络传输不同语言和编码的文字，使得双字节的 Unicode 能够在现存的处理单字节的系统上正确传输。

UTF-8 使用可变长度字节来储存 Unicode 字符，例如 ASCII 字母继续使用 1 字节储存，重音文字、希腊字母或西里尔字母等使用 2 字节来储存，而常用的汉字就要使用 3 字节。辅助平面字符则使用 4 字节。

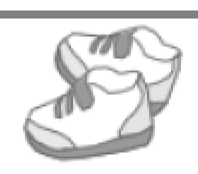
UTF-8 有两大优点：

（1）128 以下编码和单字节处理软件兼容。

（2）UTF-8 的多字节编码没有部分字节混淆问题。例如删除半汉字后整行乱码的问题在 UTF-8 里是不会出现的；任何一个字节的损坏都只影响对应的那个字符，其他字符都可以完整恢复。

脚下留神：

在开发项目的过程中，笔者建议使用 UTF-8 编码，因为该编码在不同的计算机之间使用网络传输不同语言和编码的文字时，不会出现乱码的问题。



上机演练

上机演练 1　设计 GB2312 编码格式的网页

设置页面编码格式为 GB2312 编码，运行效果如图 17.1 所示。在页面中打印出明日企业网的页面编码的提示信息，从运行结果可以知道该页面所使用的编码为 GB2312 编码。

上机演练 2　设计 GBK 编码格式的网页

设置网页的页面编码为 GBK 格式，运行效果如图 17.2 所示。在页面中打印出咖啡小屋的页面编码。



图 17.1　打印明日企业网的页面编码



图 17.2　打印咖啡小屋的页面编码

上机演练 3　设计 UTF-8 编码格式的网页

设置网页页面为 UTF-8 编码，运行效果如图 17.3 所示。在页面中打印出企业邮局收发系统所使用的页面编码。

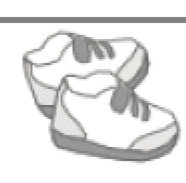


图 17.3　打印企业邮局收发系统所使用的页面编码

17.2　PHP 网页的字符编码

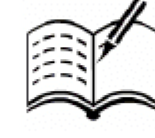
视频讲解：配套资源\mr\17\video\PHP 网页的字符编码.exe

在通过 PHP 语言开发 Web 项目时，对网页编码格式的选择非常重要，它关系着项目是否



能够接受各种不同环境的考验。例如，项目编码格式采用 GB2312，那么当项目被修改为繁体中文编码时就会出现乱码的问题，而如果最初使用 UTF-8 编码，那么无论做怎样的修改，项目的输出都不会出现乱码的问题。







17.2.1 设置编码格式

在 HTML/XHTML 页面的 head 区域中，meta 是重要标签之一。其中，meta 的 HTTP-EQUIV 关键字用于描述网页使用的字符集。常用的 HTTP-EQUIV 类型有：Content-Type 和 Content-Language（显示字符集的设定）。

这两个标签说明网页使用的语言，浏览器会据此来调用相应的字符集显示内容。例如：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

该 meta 标签是比较重要的，它指定该 HTML 页面所使用的字符集为 UTF-8，浏览器会按它指定的编码显示，包括表单提交时的字符集编码。

Content-Type 的 Content 值除了可以是“text/html”，还可以声明为“text/xml”，表示该文件是 XML 文档。

如果其中的“charset=utf-8”替换成“charset=BIG5”，则表示页面所用的字符集就是繁体中文 BIG5 码，首先应该确定该网页确实是 BIG5 编码，否则将出现乱码。

常用的 Charset 有如下几种：ISO-8859-1（英文）、BIG5（中国台湾地区使用的繁体中文）、UTF-8、Shift-JIS（日语）、x-eucjp（日语）、euc-kr/koi8-r（韩语）、US-ASCII（ASCII 基本字符集）、GB2312/EUC-CN、GBK/CP936 等字符集设置。

另外，在 HTML 页中，meta http-quiv="Content-Language"表示网页使用的语言，它的 Content 声明可以是：EN、FR、zh_CN、zh_TW 等语言代码，用来告诉搜索引擎利用此标签分类文件语言，例如，告知搜索引擎文件使用的是中文：

```
<meta http-equiv="Content-Language" content="zh_CN">
```

在 HTML 中，通过 meta 标签设置页面的编码格式，其代码如下：

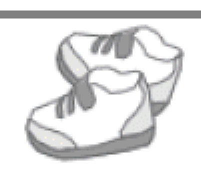
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
</head>
<body>
</body>
</html>
```

文件编码指的是文件（.html、.php）本身以何种编码形式保存。在不同的编辑工具中默认创建的文件编码格式是不同的。

17.2.2 转换编码格式

在 PHP 中，如果函数返回值的编码与页面设置的编码不同，那么就需要对函数返回值的







编码格式进行转换。转换编码可以通过 iconv()函数和 mb_convert_encoding()函数来实现。

1. 通过 iconv()函数实现编码转换

iconv()函数，转换被指定字符串的编码格式。其语法如下：

```
iconv ( string in_charset, string out_charset, string str )
```

将指定的字符串 str 由 in_charset 编码格式转换成 out_charset 编码格式，成功返回转换后的字符串；失败返回 False。

如果在参数 out_charset 后添加参数//IGNORE，那么将忽略转换时的错误；如果在出现错误时没有使用参数//IGNORE，那么将导致出错字符串后面的所有字符串无法保存。

用 iconv()函数实现 GB2312 到 UTF-8 的编码转换，代码如下：

```
iconv("GB2312","UTF-8",$text);
```

用 iconv()函数实现 UTF-8 到 GB2312 的编码转换，代码如下：

```
iconv("UTF-8","GB2312",$text);
```

iconv()函数的另一种格式是使用“//IGNORE”参数，代码如下：

```
iconv("UTF-8","GB2312//IGNORE ",$text);
```

IGNORE 的作用是忽略转换时的错误，有时 iconv()函数在转换字符“—”到 GB2312 编码时会出错，如果没有 IGNORE 参数，则该字符后面的所有字符串都无法被保存。

2. 通过 mb_convert_encoding()函数实现编码转换

mb_convert_encoding()函数，实现字符编码的转换。其语法如下：

```
mb_convert_encoding ( string str, string to_encoding [, mixed from_encoding] )
```

mb_convert_encoding()函数将字符串 str 的编码由 from_encoding 转换为 to_encoding。如果指定多个 from_encoding 编码，则使用逗号进行分隔；如果没有指定 from_encoding 编码，则使用内在的编码。

mb_convert_encoding()是 mbstring 扩展库中的函数，相对于 iconv()函数而言，mb_convert_encoding()函数的执行速度更快。例如：

用 mb_convert_encoding()函数实现 GB2312 到 UTF-8 的编码转换，代码如下：

```
mb_convert_encoding($text,"UTF-8","GB2312");
```

脚下留神：

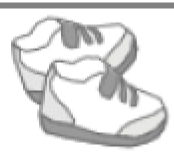
由于 mb_convert_encoding()函数属于 mbstring 扩展库，所以要应用 mb_convert_encoding()函数，必须加载 mbstring 扩展库。即在 php.ini 文件中定位到“;php_mbstring.dll”，将其前面的分号“;”去掉，保存并重新启动 Apache 服务器。

例 17.1 为了更好地理解和应用 PHP 中的编码转换函数，下面编写一个实例，分别应用 iconv()函数和 mb_convert_encoding()函数，实现由 GB2312 编码到 UTF-8 编码的转换。（**实例位置：配套资源\mr\17\example\17.1**）

（1）通过 Dreamweaver 新建一个动态 PHP 文件，设置网页默认编码格式为 GB2312。

（2）编写 PHP 脚本，定义两个字符串变量，并且通过 iconv()和 mb_convert_encoding()函数完成对字符串由 GB2312 编码到 UTF-8 编码格式的转换。其代码如下：

```
<?php
$texts="明日科技";                                    //定义字符串
```



```
$strs=iconv("gb2312","utf-8",$texts);          //将GB2312编码转换成UTF-8编码
echo $strs;                                    //输出字符串
echo "<br>";
$text="编程词典";                              //定义字符串
$str=mb_convert_encoding($text,"UTF8","gb2312");  //将GB2312编码转换成UTF-8编码
echo $str;
?>
```

运行本实例，由于在 GB2312 编码格式的页面中输出 UTF-8 编码的字符串，所以导致输出字符串乱码，如图 17.4 所示。

此时，在 IE 浏览器中，单击“查看”/“编码”/Unicode（UTF-8），页面则设置为 UTF-8 编码格式，内容则可以正常输出，如图 17.5 所示。



图 17.4　编码格式不匹配导致输出内容乱码



图 17.5　设置正确的编码格式后输出的内容

17.2.3　检测字符串的编码

在 PHP 中，可以通过 mbstring 扩展库中提供的 mb_detect_encoding()函数来检测字符串使用的是何种字符编码。

mb_detect_encoding()函数，用于检测编码的格式。其语法如下：

```
mb_detect_encoding ( string str [, mixed encoding_list [, bool strict]] )
```

mb_detect_encoding()函数用于检测字符串 str 的编码格式，返回值为数据流的代码页（Code Page）。如果使用了参数 encoding_list，那么必须使用逗号进行分隔；如果省略了 encoding_list 参数，那么将使用默认的编码。

多学两招：

什么是代码页（Code Page）？

代码页（Code Page）可以被理解为计算机的内码，GB2312 对应的 Code Page 是 EUC-CN，GBK 对应的 Code Page 是 CP936。

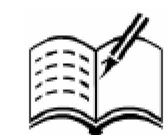
在操作系统中，使用代码页来满足不同国家和地区的文字使用需求。在简体中文的 Windows 下，默认保存文件的编码为 ANSI 或默认编码，对应的代码页是 EUC-CN。如果保存的编码是 UTF-8，那么显示也是 UTF-8，所以可以得到来源数据的编码格式。

例如，通过 mb_detect_encoding 函数检测程序中指定字符串的编码格式，其代码如下：

```
<?php
$str="获取源文件的编码格式";                   //定义变量
echo mb_detect_encoding($str)."\n";            //输出编码格式
?>
```

运行结果为：UTF-8





多学两招：

浏览器是如何区分字符编码的优先级的？

由于字符编码定义的方法很多，所以当使用多种方式时就存在一个优先级的问题。下面给出浏览器在判断一个文本的解码方式时的优先级别（由高到低）。

（1）HTTP 消息报头的 Content-Type 字段中的 charset 参数。

（2）通过 meta 标签声明，将 http-equiv 设置成 Content-Type。

（3）一些元素的 charset 属性设置。

除了这些设置外，用户浏览器也可以自发的试探，例如一些浏览器默认的字符编码方式是 ISO-8859-1，如果没有定义这些设置，浏览器就会试着使用 ISO-8859-1 来解码文本。

除此之外，用户也可以手动改变编码。一般来说，浏览器的菜单栏中都会列出不同的字符编码方式，用户可以从中选择一个作为试探，如果文本显示正确就证明文本采用了这种编码方式；如果文本显示不正确，那么可以选择其他编码继续探索，直到文本显示正确为止。

上机演练

上机演练 4　通过 iconv()函数实现编码格式的转换

在 Dreamweaver 开发工具中创建一个 UTF-8 编码格式的网页，应用 PHP 中的 setlocale()函数完成区域化设置，输出美式的系统当前时间和中文的系统当前时间。在输出中文的系统当前时间时，因为 strftime()函数的返回值为 GB2312 编码，而当前页面使用的是 UTF-8 编码格式，所以直接输出将会出现乱码，如图 17.6 所示。

所以必须应用编码格式转换函数对它的返回值进行编码格式转换才能够正常输出中文格式的时间，这里应用 iconv()函数对 strftime()函数的返回值进行编码格式转换，转换后的运行结果如图 17.7 所示。



图 17.6　输出中文时间乱码



图 17.7　正常输出中文时间

17.3　PHP 开发中的乱码问题

视频讲解：配套资源\mr\17\video\PHP 开发中的乱码问题.exe

PHP 开发中编码格式的设置并没有想象的那么复杂，只要保持文档的实际编码与编码指令一致，那么基本上就不会出现乱码的问题。下面对开发中出现乱码的问题进行分析。

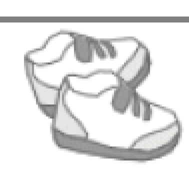
17.3.1　解决页面中的乱码问题

如果在设计的页面中出现乱码，那么可以使用下面的两种方法来解决。

（1）使用 meta 标签设置页面编码。

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

meta 标签的作用是声明客户端的浏览器用什么字符集编码显示该页面，其中 charset 可以







设置为 GB2312、GBK、UTF-8（和 MySQL 不同，MySQL 是 UTF-8）等。大部分的页面都是通过这种方式来告诉浏览器显示该页面时采用什么编码。

（2）通过 header()函数设置页面的编码格式。

```
header("content-type:text/html; charset=utf-8");
```

header()函数的作用是把括号内的信息发到 http 标头。通过 header()函数设置 charset 的值来指定页面的编码格式，其作用与 meta 标签相同。但不同之处是，如果同时使用 header()函数和 meta 标签来设置页面的编码格式，浏览器会采用 header()函数指定的编码格式，而不会使用 meta 标签设置的编码的格式，这就是 header()函数的特殊之处。header()函数多数应用在纯 php 的文件中，通过它来设置文件的编码格式。

例 17.2　通过 header()函数和 meta 标签，分别为页面设置 UTF-8 编码和 GB2312 编码，输出一个中文字符串，看一下最终的输出结果是应用的哪种编码格式。**（实例位置：配套资源\mr\17\example\17.2）**

（1）通过 Dreamweaver 新建一个动态 PHP 文件，通过 header 区域中 meta 标签的 HTTP-EQUIV 属性，设置网页编码格式为 GB2312。

（2）编写 PHP 脚本，通过 header()函数设置页面编码格式为 UTF-8。

（3）编写 PHP 脚本，通过 echo 语句输出一个中文字符串。代码如下：

```
<?php
header("content-type:text/html;charset=UTF-8");                //设置编码格式为UTF-8
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>比较编码格式的设置</title>
</head>
<body>
<?php
echo "这天太冷了";
?>
</body>
</html>
```

运行结果如图 17.8 所示。由于页面中使用了 UTF-8 编码格式，所以导致输出内容为乱码。



图 17.8　在 UTF-8 编码格式的页面中输出 GB2312 编码格式的字符串

脚下留神：

header()函数只能在 PHP 脚本中使用，并且通过该函数设置页面为 UTF-8 编码时，使用的格式是 UTF-8。



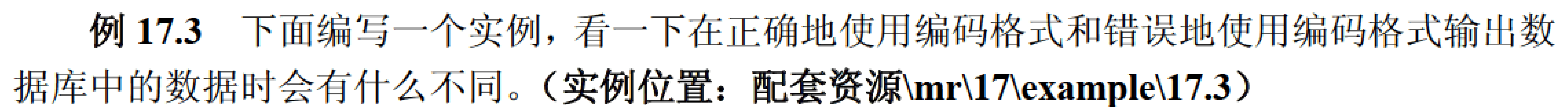
17.3.2　数据库中的字符集编码问题



通过 PHP 操作数据库中的数据时，如果网页中使用的是 UTF-8 编码，那么就可以在连接数据库的文件中执行 mysql_query("SET NAMES UTF8")语句，指定数据库的编码为 UTF-8，这样再读取出的中文数据就不会出现乱码的问题。也就是说，保证页面编码格式与数据库中输出数据的编码格式一致就不会出现乱码的问题。同样，在向数据库中添加数据时，也要保证编码格式的统一。

例 17.3　下面编写一个实例，看一下在正确地使用编码格式和错误地使用编码格式输出数据库中的数据时会有什么不同。（**实例位置：配套资源\mr\17\example\17.3**）

（1）创建 db_database17 数据库，并创建一个包含 4 个字段的 tb_charset 数据表，设置数据库字符集为 UTF-8。创建数据表的完整 SQL 语句如下：



```
SQL 查询:
CREATE TABLE `tb_charset` (
  `id` INT( 10 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  `user` VARCHAR( 80 ) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL ,
  `pass` VARCHAR( 80 ) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL ,
  `address` VARCHAR( 80 ) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULI
) ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

图 17.9　创建数据表的 SQL 语句

（2）创建 conn.php 文件，完成与 db_database17 数据库的连接，并且通过 mysql_query()函数设置数据库的编码格式为 UTF-8。

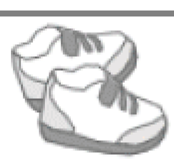
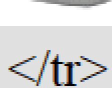
```
<?php
$conn=mysql_connect("localhost","root","111") or dir("服务器连接失败".mysql_error());//连接数据库服务器
mysql_select_db("db_database17",$conn) or dir("数据库连接失败".mysql_error());//连接db_database17
mysql_query("set names utf8");                              //设置数据库编码格式
?>
```

脚下留神：

在通过 mysql_query()函数设置数据库的字符集时，如果要设置数据库的字符集为 UTF-8，那么使用的格式是 UTF-8。它与通过 header()函数设置字符集为 UTF-8 是不同的。

（3）通过 Dreamweaver 新建一个动态 PHP 文件，通过 header 区域中 meta 标签的 HTTP-EQUIV 属性设置网页编码格式为 UTF-8。载入数据库连接文件，完成 MySQL 数据库中数据的循环输出。其关键代码如下：

```
<?php
    include_once("conn/conn.php");                           //载入数据库连接文件
    $query=mysql_query("select * from tb_charset",$conn);    //执行查询语句
    while($myrow=mysql_fetch_array($query)){                 //循环输出查询结果
?>
  <tr>
      <td align="center" bgcolor="#FFFFFF"><?php echo $myrow['id'];?></td>
    <td height="17" align="center" bgcolor="#FFFFFF"><?php echo $myrow['user'];?></td>
    <td bgcolor="#FFFFFF"><?php echo $myrow['pass'];?></td>
```

```
    </tr>
    <?php }?>
```

运行结果如图 17.10 所示。此时设置的数据库字符集与页面编码格式是相同的，所以内容可以正确的输出。

但是，如果在 conn.php 中将数据库字符集设置为 GB2312，那么再次运行本程序时，将输出图 17.11 所示的内容，中文字符串将出现乱码。如果在 conn.php 中将设置数据库字符集的操作注释掉，那么在运行本程序时输出的中文字符串同样会出现乱码。

ID	用户名	密码
1	明日	123456
2	科技	456789

图 17.10　正确地读取数据库中的数据

ID	用户名	密码
1	÷□	123456
2	?□?	456789

图 17.11　编码格式不统一导致输出内容乱码

17.3.3　避免截取中文字符串时出现乱码

这里向读者推荐 mb_substr()函数，通过它来对中文字符串进行截取，就可以避免在截取中文字符串时出现乱码。

mb_substr()函数，对字符串进行截取操作，并且支持中文字符串的截取。其语法如下：

```
string mb_substr ( string str, int start [, int length [, string encoding]] )
```

mb_substr()函数的参数说明如表 17.1 所示。

表 17.1　mb_substr()函数的参数说明如下

参　数	说　明
str	必选参数，指定操作的字符串
start	必选参数，指定截取的开始位置。参数 start 的指定位置是从 0 开始计算的，即字符串中的第一个字符表示为 0
length	指定截取的字符串长度
encoding	设置字符串的编码格式

例 17.4　为了更好地理解 mb_substr()函数的应用，下面编写一个实例，应用 mb_substr()函数对字符串进行截取，并且输出截取后的字符串。**（实例位置：配套资源\mr\17\example\17.4）**

首先，定义字符串变量$str，变量值为“PHP 自学手册”。然后，通过 substr()函数对字符串进行截取，从第一个字节开始，截取 6 个字节。最后，通过 mb_substr()函数对字符串进行截取，同样从第一个字节开始，截取 6 个字节。其关键代码如下：

```
<?php
$str="PHP  自学手册";                    //定义字符串变量
$m_sub=mb_substr($str,0,6,"UTF-8");    //截取6个字节，并且设置返回字符串的编码格式为UTF-8
echo $m_sub."<br>";                    //输出结果
?>
```

运行结果为：PHP 自学

通过 mb_substr()函数截取字符串时，一个英文字符串即为一个字节，而一个中文汉字同样也定义为一个字节，这样就避免了在截取中文字符串时出现乱码的问题。这也就是在本实例中文为什么通过 mb_substr()函数对字符串“PHP 自学手册”进行截取时返回“PHP 自学”这样一个值的原因。它将“PHP”定义为 3 个字节，空格为一个字节，“自学”为两个字节。



Note

上机演练

上机演练 5　论坛中控制帖子标题输出的长度

在论坛中输出帖子的标题时，为了确保页面的美观和完整，必须控制超长标题的输出，并以省略号来替换，此时就需要对帖子的标题进行截取，这里推荐读者使用 mb_substr()函数对帖子的标题进行截取，可以避免截取中文时出现乱码的问题。本例实现论坛中帖子标题的循环输出，并且对超长的标题进行截取，同时使用省略号进行替换，运行效果如图 17.12 所示。



ID	标题	时间
19	致明日科技	2009-11-16 09:40:40
15	PHP编程...	2009-09-04 10:42:53
23	PHP编程...	2010-08-19 03:12:22
24	PHP编程...	2010-08-19 03:13:31

图 17.12　论坛中控制帖子标题输出的长度

本 章 摘 要

1．字符集和编码。

2．PHP 网页的字符串编码。详细讲解了如何在 PHP 网页中进行编码操作，如何实现字符编码的转换和字符编码的检测。

3．PHP 开发中的乱码问题。PHP 开发中经常遇到的编码问题的解决方案。同时还介绍了 PHP 的区域化设置。在开发 PHP 网页的过程中，除了在程序中使用恰当的字符编码和语言配置文件以外，熟悉 PHP 编码规则和掌握 PHP 的编码风格也是相当重要的，正确地编写 PHP 脚本可以提高代码的质量，提高程序的可维护性以及开发速度和效率。

习　　题

1．Unicode 字符集是 Universal Multiple-Octet Coded Character Set 通用多（　　）编码字符集的简称。

A．8 位　　B．16 位　　C．32 位

2．ISO8859 是由 ISO（　　）编码基础上制定的编码标准。

A．GB2312　　B．ASCII　　C．GBK　　D．UTF-8

3．下面哪些不属于 UTF-8 的优点（　　）。

A．128 以下编码和单字节处理软件兼容

B．UTF-8 的多字节编码没有部分字节混淆问题。例如删除半汉字后整行乱码的问题在 UTF-8 中是不会出现的；任何一个字节的损坏都只影响对应的那个字符，而其他字



符都可以完整恢复

C. 易于识别

4. 在程序开发中转换编码集需要使用函数（　　）。

A. iconv　　　　B. string　　　　C. is_dir

5. 下列编码可以被理解为计算机内码的是（　　）。

A. UTF-8　　　　B. GBK　　　　C. Code Page

6. 利用 meta 标签设置当前页面编码为 UTF-8 的代码为（　　）。

7. 检测字符串编码格式需要使用函数（　　）。

8. 利用 header 函数设置页面的编码格式为 UTF-8 的代码为（　　）。

9. 截取中文字符串时为解决乱码问题可以使用函数（　　）来避免。

10. 将字符串“PHP”的编码格式由 UTF-8 转换为 GBK 的代码为（　　）。





实战模拟

学完本章后，为了让大家更好地理解和掌握本章的知识，我们设计了实战模拟栏目，以此来检验大家对本章知识的掌握情况，给大家一个理论与实践相结合的机会。（说明：上机演练和实战模拟所列实例在配套资源中提供了源码，同时读者可以参考《PHP 经典编程 265 例》一书的第 16 章内容，其中对所列实例的实现方法进行了详细讲解）。

实战模拟 1　对输出的数据进行编码格式转换

PHP 不仅可以与 MySQL 数据库搭配使用，而且也可以与其他数据库进行联合应用。通过 PHP 操作 Access 数据库，完成数据库中数据的循环输出，并且通过 iconv()函数对输出数据的编码格式进行转换，运行效果如图 17.13 所示。



图 17.13　循环输出数据库中数据

实战模拟 2　通过自定义函数解决对中文字符串截取时乱码

substr()函数是按字节进行截取字符串的，在截取中文字符串时，由于一个汉字由两个字符组成，如果只截取一个字符就会出现乱码。本例通过自定义函数解决对中文字符串截取时乱码，运行结果如图 17.14 所示。

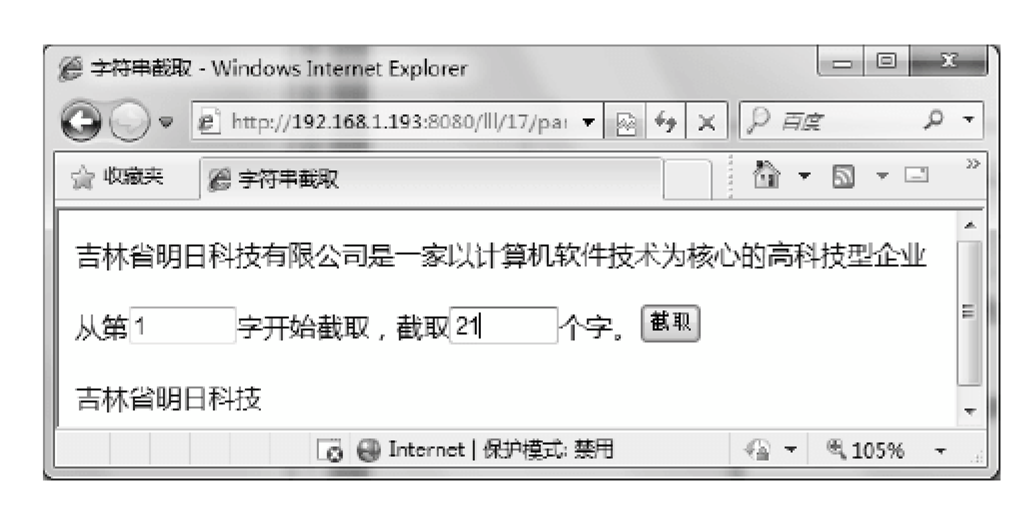


图 17.14　通过用 substr()函数解决对中文字符串截取时乱码

▶▶ 第 4 篇

实战项目篇



明日导航网

（PHP+ThinkPHP+MySQL 实现）

（自学视频、源程序：配套资源\mr\18\）

明日导航网是一个信息化管理网站。此网站的编程思想来源于 hao123 主页，提供最常用的链接，最大限度为地用户提供方便。不要小看单一的网页设计，任何基于 Web 的 B/S 架构下的程序开发都是由一个个的网页衔接而成的。程序设计的最终目标不是对某一种语言有多么深刻的理解或者实现多么复杂的功能，而是对思想的一种实现。

学习摘要：

- 基本的项目开发的思想
- 数据库的设计
- ThinkPHP 框架的架构
- ThinkPHP 控制器的应用
- ThinkPHP 模型的应用
- ThinkPHP 视图的应用
- ThinkPHP 默认模板引擎（ThinkTemplate）的应用



Note

18.1　项目设计思路

18.1.1　功能阐述

众所周知，hao123 导航网站是国内网页导航的第一品牌，是千万用户上网的第一站。此网站建于 1999 年 5 月，前名是“精彩实用网址”，后来改名为 hao123，该网站为网民提供了最便捷的上网体验，用户可以通过它快速找到自己需要的网站，而不用去记太多复杂的网址。2004 年，百度出资 5000 万人民币外加部分百度股权，收购 hao123 网站，由此可见，一个小小的 Web 网页同样可以实现很大的成就。本章模拟 hao123 网站，应用 ThinkPHP 框架，开发一个属于自己的导航网站。其目的是让读者从网站开发的实战中体会 ThinkPHP 的强大功能。

18.1.2　功能结构

本网站包括前台和后台两大功能模块，其具体功能结构如图 18.1 所示。

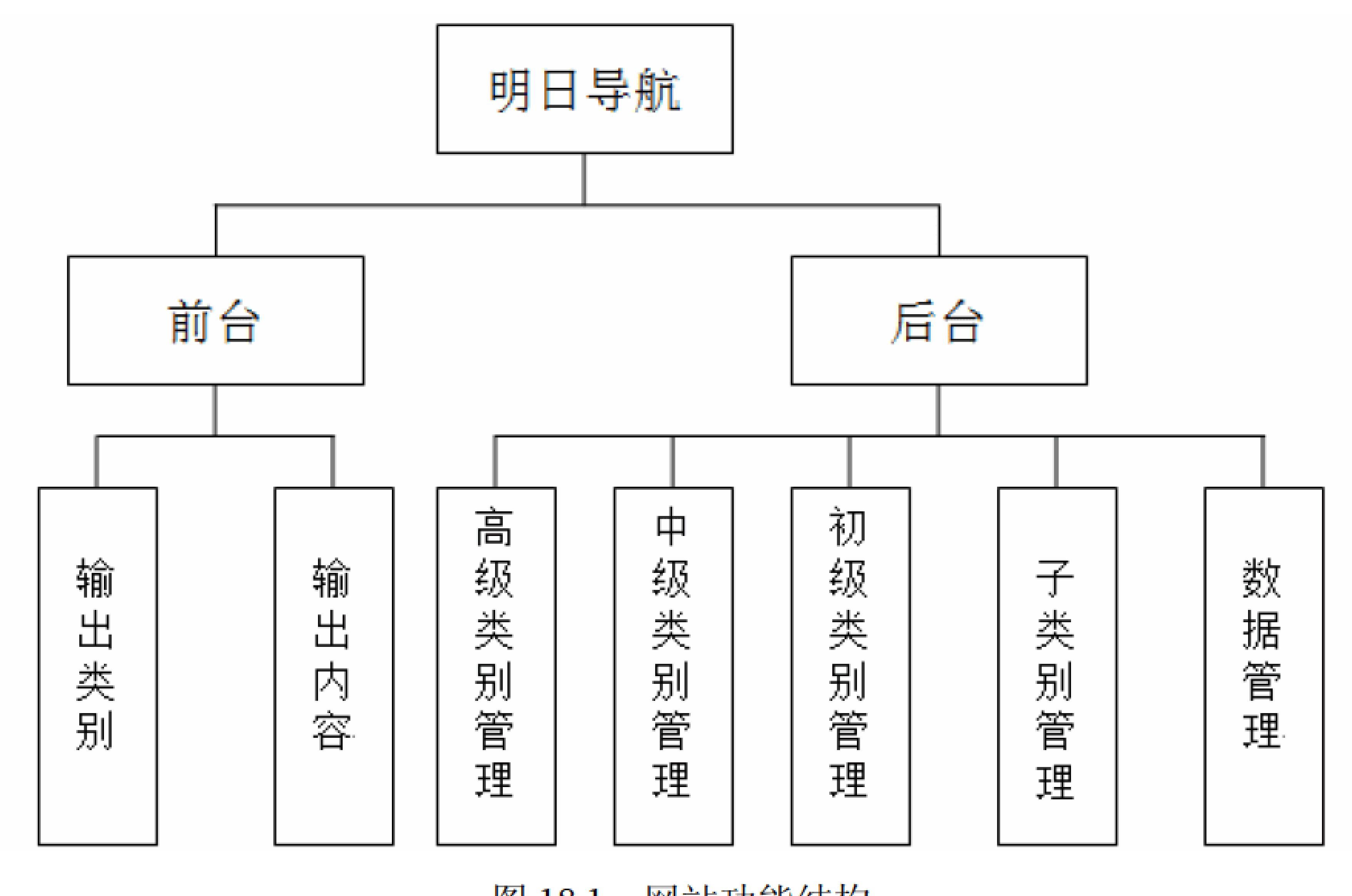


图 18.1　网站功能结构

18.1.3　系统预览

明日导航网由多个页面组成，下面具体进行介绍。

（1）网站主页面的运行效果如图 18.2 所示，主要功能是对各种网站网址进行整合输出，以达到方便浏览者使用的目的。

（2）网站中级类别页面中展示更多的资源信息，其运行效果如图 18.3 所示。

（3）明日导航网站后台登录地址是 http://127.0.0.1/MR/18/admin.php。后台登录用户名为 mr，密码为 mrsoft，登录页面如图 18.4 所示。



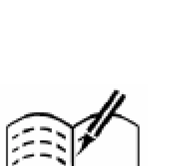




图 18.2　明日导航网主页



图 18.3　网站中级类别页面中展示更多的资源信息



图 18.4　明日导航网站后台登录页面

（4）明日导航后台管理主页完成对数据信息、类别信息的添加、删除和分页浏览的功能。其运行效果如图 18.5 所示。



图 18.5　明日导航后台管理主页面





18.2　数据库设计

视频讲解：配套资源\mr\18\video\数据库设计.exe

18.2.1　数据库设计

在明日导航网中采用的是 MySQL 数据库，用来存储各种网站的链接、名称等信息，并且通过类别数据表对各种网站进行分类。这里将数据库命名为 db_database18，其中包含的数据表如图 18.6 所示。

服务器: localhost ▸ 数据库: db_database18

表	类型	整理	说明
a_common	MyISAM	utf8_general_ci	链接数据表
a_elementarytype	MyISAM	utf8_unicode_ci	初级类别表
a_hightype	MyISAM	utf8_unicode_ci	高级类别表
a_middletype	MyISAM	utf8_unicode_ci	中级类别表
a_smalltype	MyISAM	utf8_unicode_ci	子类别表

图 18.6　数据库结构

18.2.2　数据表设计

根据设计好的 E-R 图在数据库中创建数据表。下面给出数据表结构。

1．常用链接信息表（a_common）

常用链接信息表用于存储常用链接的相关信息，其结构如表 18.1 所示。

表 18.1　常用链接信息表（a_common）

字　段	类　型	额　外	说　明
id	int（4）	auto_increment	链接 ID
highid	int（4）		高级类别 ID
middleid	int（4）		中级类别 ID
elementaryid	int（4）		初级类别 ID
smallid	int（4）		子类别 ID
title	varchar（100）		链接名称
href	text		链接网址

2．初级类别信息表（a_elementarytype）

初级类别信息表用于存储中级类别下对应的初级类别名称，其结构如表 18.2 所示。

表 18.2　初级类别信息表（a_elementarytype）

字　段	类　型	额　外	说　明
id	int（4）	auto_increment	初级类别 ID，主键
middleid	int（4）		中级类别 ID
EnglishName	varchar（80）		类别的英文名称
ChineseName	varchar（80）		类别的中文名称

3．高级类别信息表（a_hightype）

高级类别信息表用于存储导航网站中设置的高级类别分类信息，其结构如表 18.3 所示。

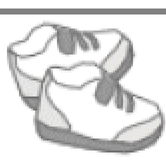
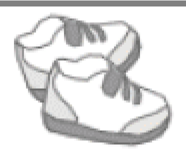
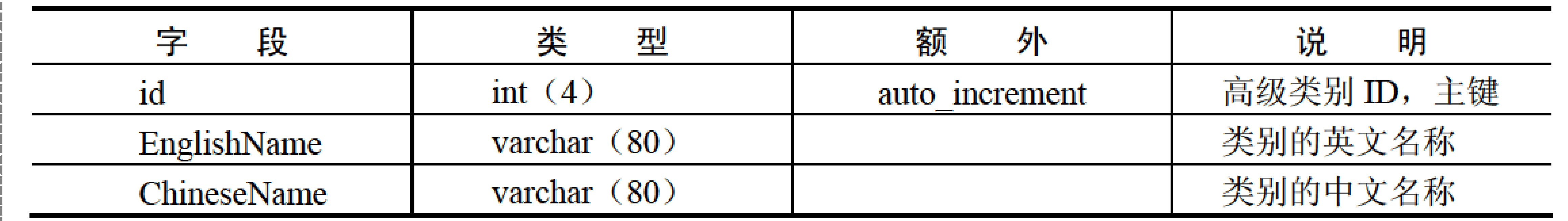



表 18.3　高级类别信息表（a_hightype）



字　段	类　型	额　外	说　明
id	int（4）	auto_increment	高级类别 ID，主键
EnglishName	varchar（80）		类别的英文名称
ChineseName	varchar（80）		类别的中文名称



4．中级类别信息表（a_middletype）

中级类别信息表用于存储中级类别分类信息，其结构如表 18.4 所示。

表 18.4　中级类别信息表（a_middletype）

字　段	类　型	额　外	说　明
id	int（4）	auto_increment	中级类别 ID，主键
highid	int（4）		高级类别 ID
EnglishName	varchar（80）		类别的英文名称
ChineseName	varchar（80）		类别的中文名称

18.2.3　连接数据库

在应用 ThinkPHP 框架开发的项目中，前后台连接数据库操作的文件分别存储于 18\Home\Conf 和 18\Admin\Conf 文件夹下，名称为 config.php。其关键代码如下：

```
<?php
return array(
    'DB_TYPE' => 'mysql',                //设置数据库类型
    'DB_HOST' => 'localhost',            //设置数据库服务器
    'DB_USER' => 'root',                 //设置用户名
    'DB_PWD' => '111',                   //设置数据库密码
    'DB_NAME' => 'db_database18',        //指定连接的数据库
    'DB_PREFIX' => 'a_',                 //设置数据表名称前缀
);
?>
```

18.3　ThinkPHP 架设项目结构

视频讲解：配套资源\mr\18\video\ThinkPHP 架设项目结构.exe

18.3.1　下载 ThinkPHP 框架

ThinkPHP 是一个免费开源、快捷、简单的 OOP 轻量级 PHP 开发框架。它遵循 Apache 2 开源协议发布，是为了敏捷的企业级开发而诞生的。获取 ThinkPHP 的方式有很多。

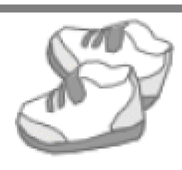
官方网站为：http://thinkphp.cn。

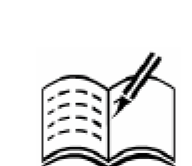
SVN 的下载地址为：

完整版本 http://thinkphp.googlecode.com/svn/trunk。

核心版本 http://thinkphp.googlecode.com/svn/trunk/ThinkPHP。

建议初学者下载完整版本，因为在完整版本中包括 ThinkPHP 的扩展、示例和文档，而核





Note

心版本中只包括 ThinkPHP 框架，不包含扩展、示例和文档。

本项目中采用的是完整版本，下载后将其存储于项目根目录 18 之下。

18.3.2　自动生成项目目录

在载入 ThinkPHP 框架之后，首先在项目的根目录下编写入口文件。本项目中包含两个入口文件：一个是 index.php 前台入口文件；另一个是 admin.php 后台入口文件。index.php 的代码如下：

```
<?php
define('THINK_PATH', './ThinkPHP');            //定义 ThinkPHP 框架路径（相对于入口文件）
define('APP_NAME', 'Home');                    //定义项目名称
define('APP_PATH', 'Home');                    //定义项目路径
require(THINK_PATH."/ThinkPHP.php");           //加载框架入口文件
App::run();                                    //实例化一个网站应用实例
?>
```

Admin.php 文件的代码如下：

```
<?php
define('THINK_PATH', './ThinkPHP');            //定义 ThinkPHP 框架路径（相对于入口文件）
define('APP_NAME', 'Admin');                   //定义项目名称
define('APP_PATH', 'Admin');                   //定义项目路径
require(THINK_PATH."/ThinkPHP.php");           //加载框架入口文件
App::run();                                    //实例化一个网站应用实例
?>
```

然后在项目的根目录下创建前台项目文件夹 Home，创建后台项目文件夹 Admin。

接着在项目的根目录下创建 Public 文件夹，然后在 Public 文件夹下分别创建 CSS 样式文件夹、images 图片文件夹、JS 脚本文件夹和 Soft 软件存储文件夹。

最后，运行项目的前后台入口文件，自动生成前后台项目目录。

至此，应用 ThinkPHP 框架架设项目结构基本完成，其生成的项目结构如图 18.7 所示。

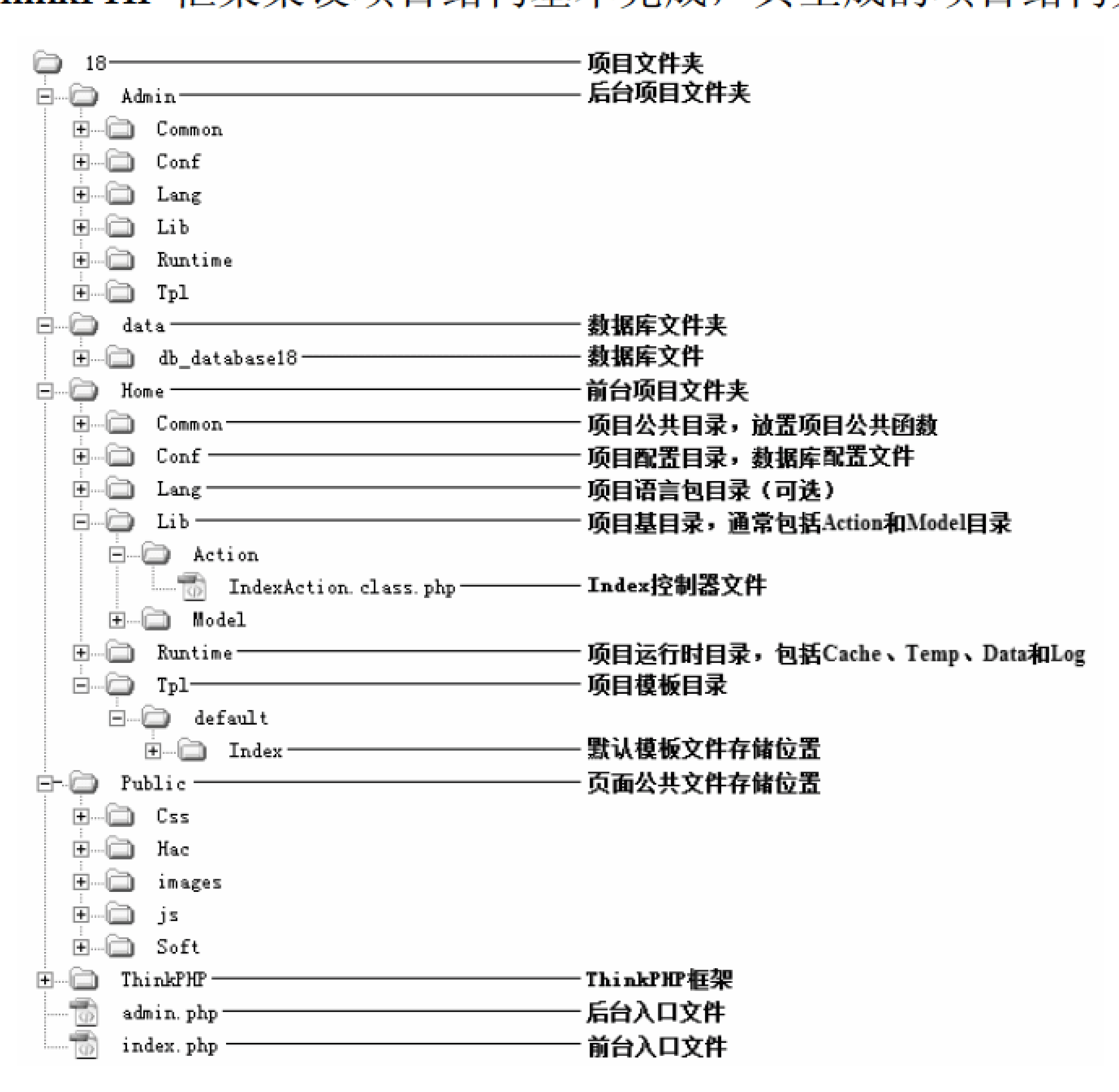


图 18.7　生成的项目结构



18.4 明日导航前台页面设计



视频讲解：配套资源\mr\18\video\明日导航前台页面设计.exe

18.4.1 页面设计概述

明日导航前台页面的功能是对本网站提供的各种信息网站进行分类输出，为浏览者查询信息提供最快捷的路径。其总体分类结构如下：生活服务、娱乐休闲、地方网站、其他、实用工具和游戏专区 6 个高级类别，在此基础上划分中级类别，中级类别下设初级类别，初级类别中还包含子类别。

在前台首页中，首先按照高级类别对数据进行分类，然后，展示中级类别，设置子页面展示中级类别包含的初级类别信息，最后直接展示一些常用网站的链接地址以及一些中级类别下包含的常用网站地址。其首页运行效果如图 18.8 所示。



图 18.8 首页运行效果

在子页面中，根据超链接传递的中级类别 ID，展示出中级类别下包含的初级类别网站信息。其运行效果如图 18.9 所示。

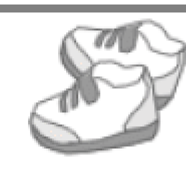




图 18.9　中级类别下数据信息的输出效果

18.4.2　控制器的创建

本项目前台控制器位于 18\Home\Lib\Action 目录下。此处创建两个控制器：IndexAction 和 MoreAction。

在 IndexAction 控制器中，定义 index 方法，查询数据库中的数据，并且将查询结果赋给指定的模板变量。其应用的技术如下：

（1）通过 M 快捷方法实例化模型类，这里包括对 middletype 和 common 两个数据表的操作。

（2）在完成类的实例化操作后，通过连贯操作完成对数据的查询，其中包括 where 语句、limit 和 select 方法。

（3）通过 assign 方法将查询结果赋给指定模板变量。

（4）通过 display 方法指定模板页。

其关键代码如下：

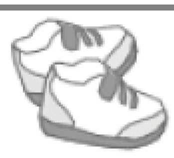
```
<?php
class IndexAction extends Action{
        public function index(){
        $middletype =M('middletype');                              //实例化模型类

        $middledata=$middletype->where('hightid=1')->select(); //查询中级类别，高级类别为“生活服务”

        $this->assign('middledata',$middledata);                   //将查询结果赋给模板变量

        $middletype=$middletype->limit('12,3')->select();          //查询中级类别
        $this->assign('middletype',$middletype);

        $com=M('common');
```




```
            $result=array();                                    //定义空数组
            for($i=0; $i<=count($middletype);$i++){             //循环输出查询结果中数据
                $search=$middletype[$i]['id'];                  //获取中级类别的 ID
                $lis=$com->where('middleid='.$search)->limit('0,7')->select(); //根据中级类别的 ID 进
行查询
                $result[]=$lis;                                 //将查询结果存储到数组中
            }
            $this->assign('listdata',$result);                  //输出中级类别数据

            $list=$com->select();                               //查询数据
            $this->assign('list',$list);                        //将查询结果赋给模板变量

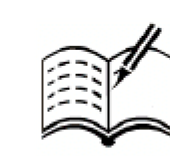
            $applieddata=$com->where('highid=5')->select();     //查询中级类别，高级类别为实用工具
            $this->assign('applied',$applieddata);
            $this->display('index');                            //指定模板页
        }
    }
    ?>
```

在 MoreAction 控制器中，定义 index、clime 和 city 3 个方法，分别用于查询指定数据表中的数据，并且将查询结果赋给指定的模板变量。其应用到的技术与 IndexAction 控制器中的相同，其关键代码如下：

```
    <?php
    class MoreAction extends Action{
        public function index(){
            $type=$_GET['link_id'];                 //获取超链接传递的 ID 值
            $ele=M('elementarytype');               //实例化模型类
            $eledata=$ele->where('middleid='.$type)->select();     //根据超链接传递的 ID 值执行查询
语句
            $com=M('common');                       //实例化模型类
            $result=array();                        //定义新数组
            for($i=0; $i<=count($eledata);$i++){ //循环读取初级类别中的数据
                $search=$eledata[$i]['id'];         //获取初级类别的 ID
                $result[]=$eledata[$i]['ChineseName'];        //将初级类别的名称存储到数组中
                $lis=$com->where('elementaryid='.$search)->select();     //根据初级类别的 ID，从
common 表中查询出数据
                $result[]=$lis;                               //将查询结果存储到数组中
            }
                $this->assign('listdata',$result);            //将数组赋给模板变量
            $this->display('index');                          //指定模板页
        }
     public function clime(){
        $type=$_GET['link_id'];                               //获取超链接传递的 ID 值
        $high=M('common');                                    //实例化模型类
        $highdata=$high->where('highid='.$type)->select(); //根据超链接传递的 ID 值执行查询语句
```



```
            $this->assign('listdata',$highdata);                //将数组赋给模板变量
            $this->display('clime');                            //指定模板页
        }
        public function city(){
            $type=$_GET['link_id'];                            //获取超链接传递的 ID 值
            $ele=M('elementarytype');                          //实例化模型类
            $eledata=$ele->where('middleid='.$type)->select(); //根据超链接传递的 ID 值执行查询语句
            $com=M('common');                                  //实例化模型类
            $result=array();                                   //定义新数组
            for($i=0; $i<=count($eledata);$i++){               //循环读取初级类别中的数据
                $search=$eledata[$i]['id'];                    //获取初级类别的 ID
                $result[]=$eledata[$i]['ChineseName'];         //将初级类别的名称存储到数组中
                $lis=$com->where('elementaryid='.$search)->select();//根据初级类别的 ID，从 common
表中查询出数据
                $result[]=$lis;                                //将查询结果存储到数组中
            }
            $this->assign('listdata',$result);                 //将数组赋给模板变量
            $this->display('city');                            //指定模板页
        }
    }
?>
```



18.4.3　视图中应用到的模板标签

在项目目录 18\Home\Tpl\default 下创建 Index 和 More 模板文件夹，分别用于存储控制器 IndexAction 和 MoreAction 对应的模板文件。在模板文件中，应用 ThinkPHP 默认模板引擎中的方法完成数据的输出和判断操作。其应用的技术如下：

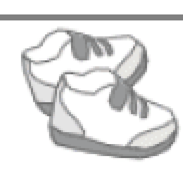
（1）通过特殊字符串的替换技术，在模板页中载入 JS 脚本、images 图片等内容。其默认的替换规则如表 18.5 所示。

表 18.5　模板中特殊字符串的替换规则

特殊字符串	替 换 描 述
../Public	被替换成当前项目的公共模板目录。通常是：/项目目录/Tpl/default/Public/
__PUBLIC__	被替换成当前网站的公共目录。通常是：/Public/
__TMPL__	替换成项目的模板目录。通常是：/项目目录/Tpl/default/
__ROOT__	替换成当前网站的地址（不含域名）
__APP__	替换成当前项目的 URL 地址（不含域名）
__URL__	替换成当前模块的 URL 地址（不含域名）
__ACTION__	替换成当前操作的 URL 地址（不含域名）
__SELF__	替换成当前的页面 URL

（2）通过 volist 标签在模板页中循环输出模板变量传递的数据。volist 标签的语法如下：

```
<volist name="list" id="vo" offset="5" length='10'>
    {$vo.name}
</volist>
```





Note

其参数说明如表 18.6 所示。

表 18.6 volist 标签的参数说明

参　数	说　明
Name	表示模板赋值的变量名称，不可随意在模板文件中改变
Id	表示当前的循环变量，可以随意指定，但确保不要和 name 属性冲突
Offset	支持输出部分数据，例如输出其中的第 5～15 条记录
Length	

Mod	控制输出记录的奇偶性，还可以控制在指定的记录换行。例如：
	//输出偶数记录
	<volist name="list" id="vo" mod="2" >
	<eq name="mod" value="1">{$vo.name}</eq>
	</volist>
	//控制一定记录的换行
	<volist name="list" id="vo" mod="5" >
	{$vo.name}
	<eq name="mod" value="4">
	</eq>
	</volist>

key	循环变量，默认设置为变量 i

多学两招：

如果要输出数组的索引，则可以直接使用 key 变量，和循环变量不同的是，key 是由数据本身决定的，而不是循环控制的，例如：

```
<volist name="list" id="vo"   >
{$key}.{$vo.name}
</volist>
```

volist 还有一个别名：iterate，其用法和 volist 相同。

（3）比较标签，在模板页中对模板变量的值进行比较操作。其语法如下：

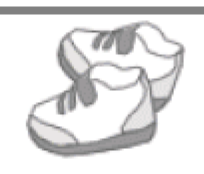
```
<比较标签  name="变量" value="值">内容</比较标签>
```

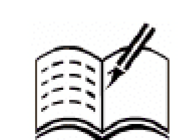
系统支持的比较标签及所表示的含义如表 18.7 所示。

表 18.7 系统支持的比较标签及所表示的含义

标　签	含　义
eq 或 equal	等于
neq 或 notequal	不等于
gt	大于
egt	大于等于
lt	小于
elt	小于等于
heq	恒等于
nheq	不恒等于

比较标签的使用方法基本相同，只是在判断的条件上有所区别。例如，要求 name 变量的





值等于 value 就输出，可以使用：

```
<eq name="name" value="value">value</eq>
```

或者

```
<equal name="name" value="value">value</equal>
```

它不但支持单条件的判断，而且还支持与 else 标签的结合应用，例如：

```
<eq name="name" value="value">相等<else/>不相等</eq>
```

比较标签中的变量可以支持对象的属性或数组，甚至可以是系统变量。例如，判断当 vo 对象的属性（或者数组，或者自动判断）等于 5 时输出，代码如下：

```
<eq name="vo.name" value="5">{$vo.name}</eq>
<eq name="vo:name" value="5">{$vo.name}</eq>
<eq name="vo['name']" value="5">{$vo.name}</eq>
```

比较标签还支持对变量使用函数。例如，判断当 vo 对象的属性值的字符串长度等于 5 时输出，代码如下：

```
<eq name="vo:name|strlen" value="5">{$vo.name}</eq>
```

变量名支持系统变量的方式，例如：

```
<eq name="Think.get.name" value="value">相等<else/>不相等</eq>
```

比较标签的比较值也支持使用变量。通常比较标签的值是一个字符串或数字，如果需要使用变量，只需要在前面添加“$”符号即可。例如，判断当 vo 对象的属性等于$a 时输出，代码如下：

```
<eq name="vo:name" value="$a">{$vo.name}</eq>
```

另外，比较标签还可以统一使用 compare 标签来进行定义。例如，判断当 name 变量的值等于 5 时输出，代码如下：

```
<compare name="name" value="5" type="eq">value</compare>
```

其中，type 属性的值就是上面列出的比较标签名称。上述写法等同于下面的表述方式。

```
<eq name="name" value="5" >value</eq>
```

其实所有的比较标签都是 compare 标签的别名。

（4）Range 标签，判断某个变量是否在某个范围之内，包括 in、notin 和 range 3 个标签。

① in 标签，判断模板变量是否在某个范围内，例如：

```
<in name="id" value="1,2,3" >输出内容 1</in>
```

② notin 标签，判断某个变量不在某个范围内，例如：

```
<notin name="id" value="1,2,3" >输出内容 2</notin>
```

标签合并使用，判断某个变量在指定范围内输出内容 1，否则输出内容 2。其语法如下：

```
<in name="id" value="1,2,3" >输出内容 1<else/>输出内容 2</in>
```

其中，Value 属性值可以是变量，变量的值可以是字符串或数组，例如：

```
<in name="id" value="$var" >输出内容 1</in>
```

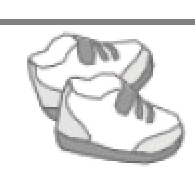
③ range 标签，替换 in 和 notin 标签，其语法如下：

```
<range name="id" value="1,2,3" type="in" >输出内容 1</range>
```

其中，type 属性的值可以是 in 或 notin。

18.4.4　在视图中创建模板文件

在控制器 18\Home\Lib\Action 目录下创建了 IndexAction 和 MoreAction 两个控制器，那么





同样在视图 18\Home\Tpl\default 目录下，创建 Index 和 More 两个模板文件夹，用于存储对应控制器的模板文件。

（1）在视图 18\Home\Tpl\default\Index 目录下，只有一个模板文件 index.html，是明日导航网站的主页，根据类别对网站提供的导航信息进行输出，并且创建子网页超链接，链接到 More 模板文件夹下的模板文件。其中应用 volist 标签循环输出控制器中查询到的中级类别数据，其关键代码如下：

```
<volist name="middletype" id="mid" key="k" >
<TR class='bg<in name="k" value="1,3,5,7,9,11,13,15,17,19,21,23,25,27,29" >1<else/>2</in>'>
    <TH   width=60><A   href="__APP__/More/index?link_id={$mid.id}">{$mid.ChineseName}
</A></TH>
    <TD class=s_widen width=636>
        <iterate name="listdata" id="child">
            <volist name="child" id="grand">
                <eq name="grand.middleid" value="$mid.id">
                    <A href="{$grand.href}">{$grand.title}</A>
                </eq>
            </volist>
        </iterate>
    </TD>
    <TD width=60><B><A href="__APP__/More/index?link_id={$mid.id}">更多 &raquo;</A>
</B></TD>
</TR>
</volist>
```

在这段代码中，应用 volist 和 iterate 标签进行嵌套，循环输出三维数组中的数据；应用 in 标签控制表格中每行的背景颜色；应用 eq 比较标签判断当中级类别中的 ID 值与超链接表（a_common）中 middleid 字段的值相等时，输出超链接表中存储的网站超链接（href）和网站名称（title）。最后创建“更多”超链接，链接到 More 控制器的 index 方法中，将中级类别对应 ID 作为参数进行传递。

（2）在视图 18\Home\Tpl\default\More 目录下包含 3 个模板文件，分别是 index.html、city.html 和 clime.html。它们与 MoreAction 控制器中定义的 3 个方法相互对应，根据控制器中查询出的数据，在模板文件中应用模板引擎中的标签完成输出的判断和输出。其应用的技术已经在 18.4.3 节中进行了详细讲解，这里不再赘述，详细内容可参考本书配套资源中的源码。

18.5 明日导航后台管理设计

视频讲解：配套资源\mr\18\video\明日导航后台管理设计.exe

18.5.1 后台管理概述

明日导航网站后台管理系统可以归纳为 3 部分内容：第 1 部分，后台登录；第 2 部分，对网站中设置的分类数据和导航链接数据进行管理；第 3 部分，退出后台管理系统。明日导航后台管理系统主页的运行效果如图 18.10 所示。





图 18.10　明日导航后台管理主页面



18.5.2　通过系统配置文件存储后台登录数据

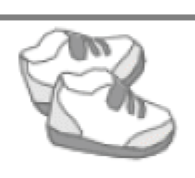
在后台登录模块中，常用的技术包括 SESSION 机制和加密技术。加密技术又分为很多种。笔者在开发本后台模块时思考了很多。是不是将管理员名称和密码统一加密保存在数据库中就安全了呢？其实并不是这样的。高明的 SQL 注入手法可以很容易地取得密文。所以，在本项目中，并没有将密码保存到数据库中，而是通过配置文件隐式地保存登录的相关信息。方法是在系统目录 Common 下，创建 PHP 脚本文件 admin.php。其代码如下：

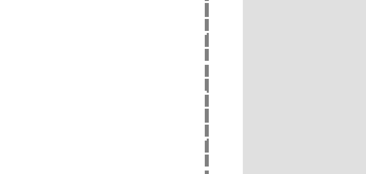
```
<?php
  Session::set("MR", "mr");                    //设置 SESSION 变量存储后台登录用户名
  Session::set("MRKJ", "mrsoft");              //设置 SESSION 变量存储后台登录密码
?>
```

这样，用户不仅可以随时随地地更改用户名和密码，而且很好地保证了密码文件的安全性。如果用户有兴趣，可以独立编写一个日志文件，用于记录 Session 的使用信息，从而实现检测非法用户暴力破解的目的。

如此存储后台管理员的登录信息后，在后台登录处理的 admin 方法中需要先载入配置文件中设置的用户名和密码，然后获取表单中提交的用户名和密码，并将其与 SESSION 变量中存储的用户名和密码进行比较，判断其是否是管理员。admin 方法中验证管理员是否登录成功的代码如下：

```
public function admin(){                    //后台登录处理方法
    Load('admin');                          //载入配置文件中设置的用户名和密码
    $username=$_POST['text'];               //获取用户名
    $userpwd=$_POST['pwd'];
    if($username==""||$userpwd==""){                        //判断用户名和密码是否为空
        $this->assign('hint','文本框内容不能为空');
        $this->assign('url','__URL__');
        $this->display('information');                      //指定提示信息模板页
    }else{
        if($username!=Session::get(MR)||$userpwd!=Session::get(MRKJ)){     //验证登录用户是否正确
            $this->assign('hint','您不是权限用户');
            $this->assign('url','__URL__/');
            $this->display('information');
        }else{
            $_SESSION['username']=$username;        //将登录用户名赋给 SESSION 变量
```





```
                    $_SESSION['userpwd']=$userpwd;
                    $this->assign('hint','欢迎管理员回归');
                    $this->assign('url',__URL__/adminIndex');    //设置后台管理主页链接
                    $this->display('information');
                }
            }
        }
```

18.5.3 后台管理架构解析

后台管理的登录从项目根目录下的 admin.php 入口文件开始，运行此文件生成后台管理项目文件夹，其具体存储于根目录下的 Admin 文件夹下。在 18\Admin\Lib\Action 目录下创建后台控制器 IndexAction，用于存储所有后台的操作方法；在 18\Admin\Tpl 目录下创建与 IndexAction 控制器对应的模板文件夹 Index，用于存储控制器中方法对应的模板文件。明日导航后台管理架构如图 18.11 所示。

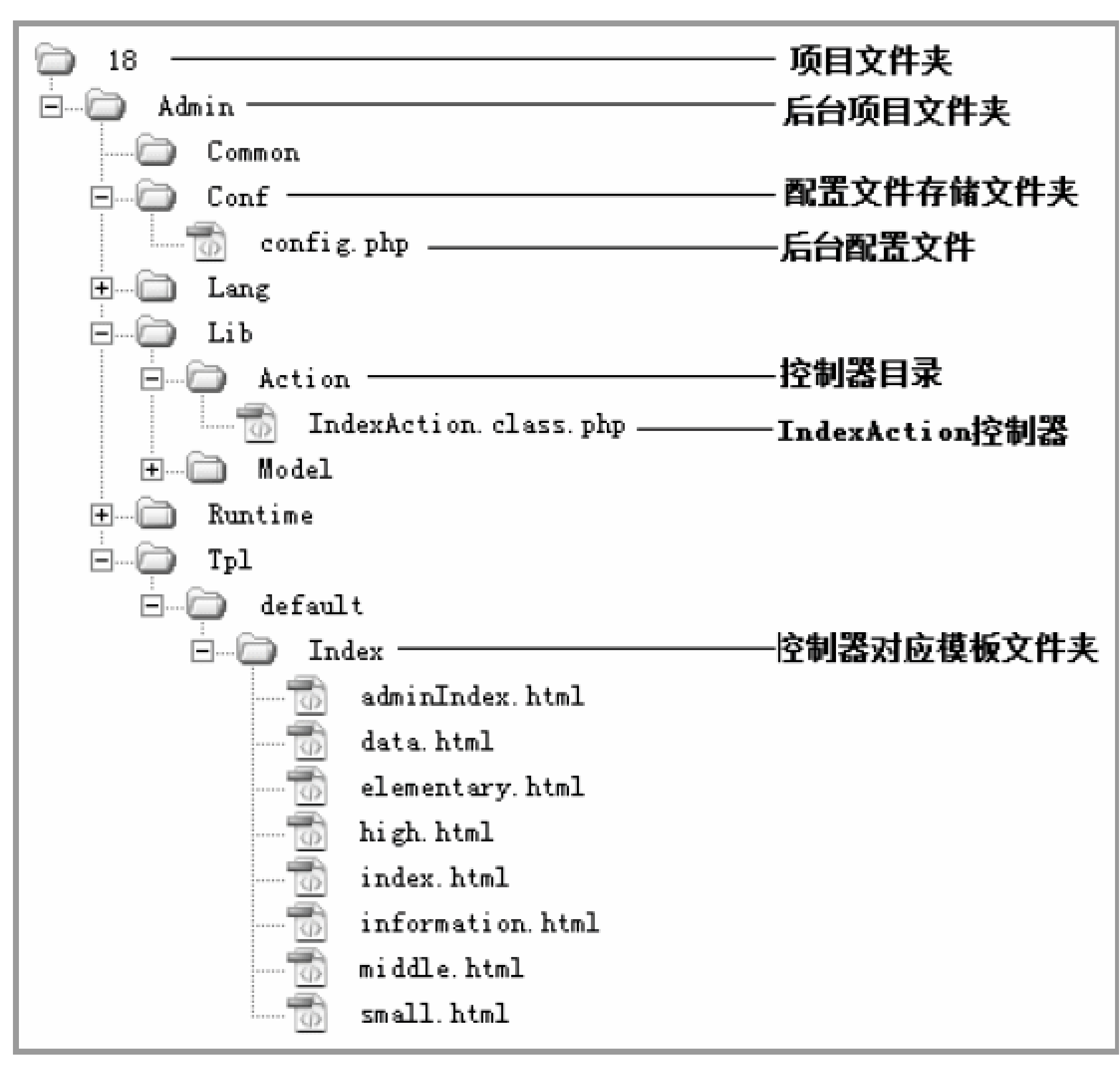


图 18.11　明日导航后台管理架构

18.5.4 ThinkPHP 框架中的分页技术

在 ThinkPHP 框架中封装了自己的分页类，其存储于 ThinkPHP 框架的 18\ThinkPHP\Lib\ORG\Util 目录下。在应用时，需要在控制器中通过 import 标签载入类文件，然后执行类的实例化操作，最后调用其中的方法完成数据的分页查询和输出。其关键代码如下：

```
①  import("ORG.Util.Page");                                  //载入分页类
        $count=$com->count();                                  //统计数据库中的记录数
②          $Page=new Page($count,8);                          //实例化分页类
③          $show= $Page->show();                              //获取分页超链接
④          $list = $com->order('id')->limit($Page->firstRow.','.$Page->listRows)->select();
                                                               //执行分页查询
        $this->assign('list',$list);                           //将分页查询结果赋给模板变量
        $this->assign('page',$show);                           //将获取的分页超链接赋给模板变量
```



① 通过 import("ORG.Util.Page");载入分页类。

② 实例化分页类，同时可以传递 3 个参数：第 1 个是页面总记录数；第 2 个是每页显示的记录数；第 3 个为可选参数，通过分页超链接的值。

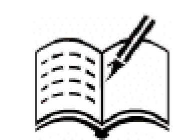
③ 调用分页类中的 show 方法，输出分页超链接。

④ Page 方法查询分页。属于新增特性，可以更加快速地进行分页查询。Page 方法的用法和 limit 方法类似，格式为：

```
Page('page[,listRows]')
```

参数 page 表示当前的页数；listRows 表示每页显示的记录数。例如 Page('2,10')，表示每页显示 10 条记录，获取第 2 页的数据。

listRow 如果为空，则会读取 limit('length') 的值，例如 limit(25)->page(3);，表示每页显示 25 条记录，获取第 3 页的数据。如果 limit 也未进行设置，则默认为每页显示 20 条记录。



指点迷津：

在通过 Page 方法进行数据的分页查询时，page 方法的第一个参数是当前的页数，需要使用$_GET['p']获取；第二个参数是当前页显示的记录数。

18.5.5　后台管理视图中应用的模板标签

在后台管理视图中应用的模板标签介绍如下。

（1）在明日导航网站的后台管理系统中，应用模板引擎中的 switch 标签创建一个简单的网页框架，在 adminindex 方法中，根据超链接传递的值实现在不同页面之间的跳转操作。

switch 标签的语法如下：

```
<switch name="变量" >
    <case value="值 1">输出内容 1</case>
    <case value="值 2">输出内容 2</case>
    <default  />默认情况
</switch>
```

（2）应用标签中的系统变量，输出超链接传递的参数值。其语法如下：

```
{$Think.get.pageNumber }
```

（3）应用 include 标签包含外部模板文件。其语法如下：

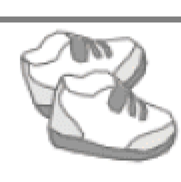
```
<include file="完整模板文件名" />             //使用完整文件名包含
<include file="操作名" />                     //包含当前模块的其他操作模板文件
<include file="模块名:操作名" />              //包含其他模块的操作模板
<include file="主题名@模块名:操作名" />       //包含其他模板主题的模块操作模板
<include file="$变量名" />                    //用变量控制要导入的模版
```

（4）通过 foreach 标签循环输出模板变量传递的数据。其语法如下：

```
<foreach name="list" item="vo" >
    {$vo.id}
    {$vo.name}
</foreach>
```

参数 name 指定在控制器中设置的模板变量名称，必须与控制器中设置的名称相同；参数 item 在 foreah 语句中自行定义的变量，用于输出模板变量传递的数据。

（5）通过 if 标签完成更加复杂的判断操作。例如，判断当变量 name 的值等于 1 或大于





Note

100 时输出 value1 的值，或者当 name 的值等于 2 时输出 value2 的值，否则输出 value3 的值。其代码如下：

```
<if condition="($name eq 1) OR ($name gt 100) ">
    value1
<elseif condition="$name eq 2" />
    value2
<else />
    value3
</if>
```

在 condition 属性中可以支持 eq 等判断表达式 ，等同于比较标签，但是不支持带有“>”、“<”等符号的用法，因为会混淆模板解析。

另外，在 condition 属性中还可以使用 PHP 代码，例如：

```
<if condition="strtoupper($user['name']) neq 'THINKPHP' ">
    ThinkPHP
<else />
    other Framework
</if>
```

多学两招：

由于 if 标签的 condition 属性中基本上使用的是 PHP 语法，所以使用判断标签和 switch 标签会显示更加简洁，原则上说，能够用 switch 和比较标签解决的问题尽量不用 if 标签解决。因为 switch 和比较标签可以使用变量调节器和系统变量。如果对于某些特殊的要求，if 标签仍然无法满足要求，可以使用原生 PHP 代码或 PHP 标签来直接书写代码。

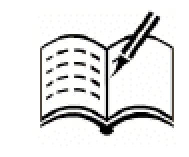
18.5.6 后台登录

前面已经对明日导航后台管理系统的架构和所涉及的技术进行了详细讲解。下面讲解后台登录模块的实现方法。后台登录模块的创建由 3 部分组成，第一部分，设置后台登录的用户名和密码，已经在 18.5.2 节中进行了详细讲解，下面讲解另外两部分。

（1）第二部分，在后台管理系统的默认视图文件 index.html 中创建表单，将管理员的用户名和密码提交到 IndexAction 控制器的 admin()方法中进行处理。创建表单的代码如下：

```
<form action="__URL__/admin" method="post">
    <tr>
        <td rowspan="3">
            <img src="__ROOT__/Public/images/login_02.jpg" width="136" height="150" alt=""></td>
      <td colspan="3" width="242" height="99" background="__ROOT__/Public/images/login_07.jpg">
            <div>用户名：<input class="user" type="text" name="text"></div><br>
        <div>密  码：<input class="pwd" type="password" name="pwd"></div><br>
            </td>
        <td>
            <img src="__ROOT__/Public/images/login_04.jpg" width="26" height="99" alt=""></td>
    </tr>
    <tr>
        <td width="106" height="41">
```






```
                <input class="buttonSub" type="submit" value=""></td>
            <td width="98" height="41">
                <input class="buttonRes" type="reset" value=""></td>
            <td colspan="2">
                <img src="__ROOT__/Public/images/login_07.jpg" width="64" height="41" alt=""></td>
        </tr>
    </form>
```

（2）第三部分，在 IndexAction 控制器中创建 admin 方法，获取表单提交的用户名和密码，与 SESSION 变量中存储的用户名和密码进行比较，判断用户提交的名称和密码是否正确。如果正确，则说明是管理员，将登录用户名和密码存储到 SESSION 变量中，在 information.html 模板页中输出“欢迎管理员回归”，并在 4 秒钟后跳转到后台管理主页；否则，可能是提交用户名或者密码为空，或者提交的用户名不正确，或者密码不正确，那么将在 information.html 模板页中输出“用户名或者密码不能为空”或者“您不是权限用户”，并在 4 秒钟后跳转到后台登录页面。admin 方法的关键代码如下：

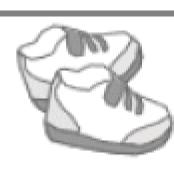
```
public function admin(){                    //后台登录处理方法
    Load('admin');                          //载入配置文件中设置的用户名和密码
    $username=$_POST['text'];               //获取用户名
    $userpwd=$_POST['pwd'];
    if($username==""||$userpwd==""){                        //判断用户名和密码是否为空
        $this->assign('hint','用户名或者密码不能为空');
        $this->assign('url','__URL__');
        $this->display('information');                      //指定提示信息模板页
    }else{
        if($username!=Session::get(MR)||$userpwd!=Session::get(MRKJ)){ //验证登录用户是
否正确
            $this->assign('hint','您不是权限用户');
            $this->assign('url','__URL__/');
            $this->display('information');
        }else{
            $_SESSION['username']=$username;            //将登录用户名赋给 SESSION 变量
            $_SESSION['userpwd']=$userpwd;
            $this->assign('hint','欢迎管理员回归');
            $this->assign('url','__URL__/adminIndex');   //设置后台管理主页链接
            $this->display('information');
        }
    }
}
```

这就是明日导航的后台登录模块，至此讲解完毕。

18.5.7　后台管理主页

管理员登录成功后，将跳转到明日导航的后台管理主页中，在后台管理主页中将根据超链接传递的参数值，实现不同子功能页面之间的跳转操作，进而实现对应的管理操作。

后台管理主页由两部分组成：第一部分，在 IndexAction 控制器中定义 adminindex()方法。





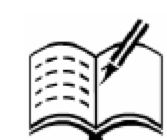
首先调用当前控制器中的 checkEnv 方法判断当前用户是否具有访问权限。然后应用 switch 语句，根据$_GET[]方法获取的超链接参数值进行判断，当参数值为 high 时，执行 IndexAction 控制器中的 high 方法；当参数值为空时，则执行默认的 common 方法。最后指定模板页 adminindex。adminindex 方法的代码如下：

```
public function adminIndex(){                //后台管理系统主页
    if(IndexAction::checkEnv()){             //判断是否具有访问权限
        switch($_GET['type_link']){          //根据超链接传递的变量值输出对应的内容
            case "high":
                IndexAction::high();         //执行 high 方法
            break;
            case "middle":
                IndexAction::middle();
            break;
            case "elementary":
                IndexAction::elementary();
            break;
            case "small":
                IndexAction::small();
            break;
            case "data":
                IndexAction::common();
            break;
            default:                         //默认输出数据管理内容
                IndexAction::common();
        }
        $this->display('adminIndex');        //指定模板页
    }
}
```

第二部分，在 18\Admin\Tpl\default\Index 模板文件夹下创建 adminindex.html 模板文件，创建后台管理中的功能导航菜单。应用 switch 标签，根据系统变量{$Think.get.type_link}获取的参数值进行判断，应用 include 标签包含不同的模板文件。其关键代码如下：

```
<img src="__ROOT__/Public/images/_html_02.gif" width="743" height="65" border="0" usemap=
"#Map" alt="">
<map name="Map">
  <area shape="rect" coords="35,24,126,55" href="__URL__/adminIndex?type_link=high">
  <area shape="rect" coords="163,21,258,53" href="__URL__/adminIndex?type_link=middle">
  <area shape="rect" coords="294,21,388,55" href="__URL__/adminIndex?type_link=elementary">
  <area shape="rect" coords="429,26,528,53" href="__URL__/adminIndex?type_link=small">
  <area shape="rect" coords="568,22,660,53" href="__URL__/adminIndex?type_link=data">
</map>
<switch name="Think.get.type_link">
    <case value="high">
        <include file="high" />
        </case>
    <case value="middle">
```






```
            <include file="middle" />
            </case>
        <case value="elementary">
            <include file="elementary" />
            </case>
        <case value="small">
            <include file="small" />
            </case>
        <case value="data">
            <include file="data" />
            </case>
        <default />
            <include file="data" />
    </switch>
```

当管理员进入后台管理主页后，单击“高级类别管理”超链接时，后台管理主页的运行效果如图 18.12 所示。



图 18.12　明日导航后台管理——高级类别管理页面

18.5.8　高级类别管理

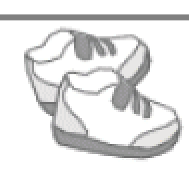
在后台管理主页中单击“高级类别管理”超链接，在主页中将分页输出高级类别数据，并且在每条记录之后都添加“删除”超链接，用于删除指定的数据。此时，如果单击主页左侧的“类别添加”超链接，将跳转到高级类别添加页面，完成高级类别的添加操作，如图 18.13 所示，如果单击“类别管理”超链接，则返回到高级类别输出的页面。

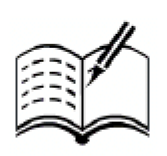


图 18.13　明日导航后台管理——高级类别添加页面

高级类别管理包括 3 个子功能，分别是数据的添加、浏览和删除。其具体实现方法如下。

（1）在 IndexAction 控制器中创建 high 方法，根据超链接传递的参数值进行判断，是执行





数据的添加操作还是执行数据的分页查询。其关键代码如下：

```
public function high(){                                   //高级类别处理方法
    header("Content-Type:text/html;charset=utf-8");       //设置编码格式
    $com=M('hightype');                                   //实例化模型类，a_hightype 表
    if($_GET['handle']=='insert'){            //判断超链接的参数值是添加语句还是管理数据
        if(IndexAction::checkEnv()){                      //判断用户是否具有添加权限
            if(isset($_POST['button'])){
                $data['ChineseName']=$_POST['ChineseName'];    //获取表单提交的数据
                $data['EnglishName']=$_POST['EnglishName'];
                $data=$com->data($data)->add();                //执行添加操作
                if($data!=false){
                    $this->assign('hint','数据添加成功！ ');
                    $this->assign('url','adminIndex?type_link=high&handle=admin');
                    $this->display('information');
                }else{
                    $this->assign('hint','添加失败！ ');
                    $this->assign('url','adminIndex?type_link=high&handle=insert');
                    $this->display('information');
                }
            }
        }
    }else{
        import("ORG.Util.Page");            //载入分页类
        $count=$com->count();               //统计总的记录数
        $Page=new Page($count,8); //实例化分页类，设置每页显示 8 条记录
        $show= $Page->show();               //输出分页超链接
        $list = $com->order('id')->limit($Page->firstRow.','.$Page->listRows)->select();     //执行
分页查询
        $this->assign('list',$list);        //将查询结果赋给模板变量
        $this->assign('page',$show);        //将获取的分页超链接赋给模板变量
    }
}
```

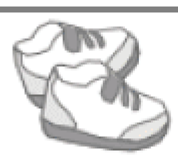
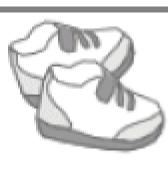
（2）在 18\Admin\Tpl\default\Index 模板文件夹下，创建 high.html 模板文件，应用 switch 标签进行条件判断，如果超链接传递的参数值是 insert，那么将输出高级类别添加的表单；如果超链接传递的是 admin，则应用 foreach 语句循环输出模板变量传递的高级类别数据，并且创建“删除”超链接，链接到 IndexAction 控制器下的 deletetype 方法，完成删除操作，以记录的 ID 值为参数值；默认输出高级类别数据。其关键代码如下：

```
<switch name="Think.get.handle">
    <case value="insert">
<form name="form1" method="post" action="__URL__/high?type_link=high&handle=insert">
<table width="750" border="1" cellspacing="1" cellpadding="1">
  <tr>
    <td colspan="2" align="center">高级类别添加</td>
  </tr>
  <tr>
```



```
<td width="178" align="right">中文名称</td>
<td width="559"><input name="ChineseName" type="text" id="ChineseName" size="40"></td>
</tr>
<tr>
<td align="right">英文名称</td>
<td><input name="EnglishName" type="text" id="EnglishName" size="40"></td>
</tr>
<tr>
<td align="center"> </td>
<td><input type="submit" name="button" id="button" value="提交">
    
<input type="reset" name="button2" id="button2" value="重置"></td>
</tr>
</table>
</form>
</case>
<case value="admin">
<table width="750" border="1" cellspacing="1" cellpadding="1">
<tr>
<td>ID</td>
<td>中文名称</td>
<td>英文名称</td>
<td>操作</td>
</tr>
<foreach name="list" item="result" >
<tr>
<td>{$result.id}</td>
<td>{$result.ChineseName}</td>
<td>{$result.EnglishName}</td>
<td><a href="__URL__/deletetype?type_link={$Think.get.type_link }&handle=
admin&link_id={$result.id}">删除</a></td>
</tr>
</foreach>
<tr>
<td colspan="4">{$page}</td>
</tr>
</table>
</case>
<default />
<table width="750" border="1" cellspacing="1" cellpadding="1">
<tr>
<td>ID</td>
<td>中文名称</td>
<td>英文名称</td>
<td>操作</td>
</tr>
```



```
                <foreach name="list" item="result" >
              <tr>
                  <td>{$result.id}</td>
                  <td>{$result.ChineseName}</td>
                  <td>{$result.EnglishName}</td>
                  <td><a href="__URL__/deletetype?type_link={$Think.get.type_link }&handle=admin&
link_id={$result.id}">删除</a></td>
                </tr>
                </foreach>
                <tr>
                  <td colspan="4">{$page}</td>
                </tr>
              </table>
        </switch>
```

（3）在 IndexAction 控制器中创建 deletetype 方法，根据超链接传递的 ID 值，执行 delete 删除语句，删除高级类别的数据。在删除高级类别中数据的同时，与其关联的中级类别、初级类别和子类别中的数据也都将被删除。其关键代码如下：

```
        function deletetype(){
            if(IndexAction::checkEnv()){                    //判断当前用户是否具备删除权限
                $cl=urldecode($_GET['link_id']);            //获取超链接传递的 ID 值
                $new=M('hightype');                         //实例化模型类
                $new=$new->execute("delete from a_hightype where id in (".$cl.")");   //以 ID 值为条件，
执行删除操作
                if($new!=false){
                    $new=M('middletype');                   //实例化中级类别表
                    $new=$new->execute("delete from a_middletype where hightid in (".$cl.")");   //删除
中级类别中的数据
                    $newe=M('elementarytype');
                    $newe=$newe->execute("delete from a_elementarytype where middleid in (".$cl.")");
                    $news=M('smalltype');
                    $news=$news->execute("delete from a_smalltype where elementaryid in (".$cl.")");
                    $this->assign('hint','数据删除成功！');
                    $this->assign('url','adminIndex?type_link=high&handle=admin');
                    $this->display('information');
                }else{
                    $this->assign('hint','出现未知错误！');
                    $this->assign('url','adminIndex?type_link=high&handle=admin');
                    $this->display('information');
                }
            }
        }
```

18.5.9 判断访问用户的权限

在明日导航后台管理系统中，为了避免其他用户登录后台管理系统给网站带来不必要的麻烦，设置了后台登录功能，只有正确登录的用户才可以对数据进行管理。那么在后台中是如何判



断用户权限的呢？其原理是：当管理员登录成功后，将其登录的用户名和密码存储到 SESSION 变量中，由此可以在执行每项操作之前，判断当前用户 SESSION 变量中存储的用户名和密码与系统指定的用户名和密码是否相同，如果相同，则具备数据的操作权限；否则，将提示“您不是权限用户”，并且跳转到管理员登录页面。



将权限判断的操作封装到 checkEnv 方法中，在该方法中完成对当前用户权限的判断操作，如果用户具备访问权限，则返回 True；否则返回 False。checkEnv 方法的语法如下：

```
public function checkEnv(){
    if($_SESSION['username']!=session::get(MR) and $_SESSION['userpwd']!=session::get(MRKJ)){
                        //判断用户名和密码是否正确
        $this->assign('hint','您不是权限用户');
        $this->assign('url','__URL__/');
        $this->display('information');
        $login=false;
    }else{
        $login=true;
    }
    return $login;        //返回判断结果
}
```

18.5.10　操作提示页面

在后台管理系统中，每执行一项操作后，无论是成功还是失败，都会跳转到同一个提示页面，返回不同的提示信息，并且跳转到指定的页面。例如，当管理员登录成功后，将弹出如图 18.14 所示的提示信息。

如果是非权限用户登录到后台，则弹出如图 18.15 所示的提示信息。

欢迎管理员回归
3秒后自动跳转，如未跳转，请单击这里

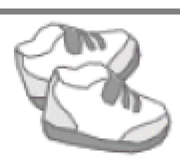
图 18.14　管理员登录

您不是权限用户
1秒后自动跳转，如未跳转，请单击这里

图 18.15　非权限用户登录

操作提示功能是根据在方法中定义的提示信息和跳转路径，经由 information.html 模板页完成提示和跳转操作的。在 information.html 模板页中，输出模板变量传递的提示信息，根据模板变量传递的路径进行跳转。其关键代码如下：

```
<table width="750" border="0" cellspacing="0" cellpadding="0" >
    <tr>
      <td align="center">{$hint}</td>
    </tr>
    <tr>
        <td align="center"><span class="spanT">5</span>秒后自动跳转，如未跳转，请单击<a
href="{$url}">这里</a></td>
      </tr>
</table>
<script type="text/javascript">
    $(function(){
        time();
```




```
        });
        var times=$("span").text();
        function time(){
            if(times==0){
                var url=$("a").attr('href');
                window.location.href=url;
            }else{
                window.setTimeout('time()',1000);
                times=times-1;
                $("span").text(times);
            }
        }
    </script>
```

项 目 发 布

1. 搭建 PHP 运行环境

由于笔者在开发项目时应用 AppServ 集成化安装包搭建 PHP 运行环境，所以建议读者也应用 AppServ 来搭建 PHP 运行环境。这样可以确保项目正常运行。

2. 具体项目发布方法

从配套资源中复制项目文件夹（例如 18），将其存储于用户本机 Apache 服务器的 Web 文件夹下。

- ☑ 这里以通过 AppServ 搭建的 PHP 运行环境为例，将程序文件夹 18 复制到 AppServ 安装后的 www 文件夹下。
- ☑ 在 18 文件夹下，查看是否存在 data 文件夹，发现本程序应用的是 MySQL 数据库，所以在运行本程序之前要完成数据库的附加。
- ☑ 将 data 文件夹下的数据库文件夹复制到用户本机 AppServ\MySQL\data 文件夹下，并且去掉数据库文件的只读属性。
- ☑ 在本网站中，前台连接数据库的操作在 18 \Home\Conf 文件夹下 config.php 文件中进行，设置服务器为 localhost、用户名为 root、密码为 111；后台连接数据库的操作在 18\Admin\Conf\config.php 文件中进行，设置服务器为 localhost、用户名为 root、密码为 111。
- ☑ 本网站后台登录网址为 http://127.0.0.1/MR/18/admin.php，用户名为 mr，密码为 mrsoft。

开 发 总 结

明日导航网运用软件工程设计思想中最流行的 MVC 设计理念，通过一个国产框架 ThinkPHP 编写而成。在本章中充分发挥 ThinkPHP 框架的作用，涉及到控制器、模型、视图以及模板引擎等方面的技术，并且对所用技术都进行了系统、详细的讲解分析。希望通过本章的学习，读者可以了解网站导航的开发流程，并且掌握 ThinkPHP 框架开发网站的流程及常用技术。

附录 A　习题参考答案

第 1 章　PHP 概述

一、选择题答案

1．ABC　2．A　3．B　4．A　5．B

二、填空题答案

1．default_timezone_set
2．windows
3．http://127.0.0.1、http://localhost
4．4
5．date

第 2 章　PHP 基础

一、选择题答案

1．C　2．ABCD　3．A　4．B　5．B

二、填空题答案

1．$b=$a;
2．.
3．4
4．0
5．define()

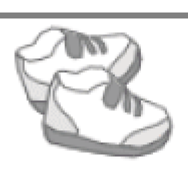
第 3 章　PHP 函数

一、选择题答案

1．A　2．A　3．A　4．B　5．D

二、填空题答案

1．echo、var_dump

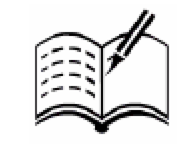


2．rmdir

3．date

4．输出变量的相关信息

5．没有任何内容输出



第 4 章　PHP 流程控制语句

一、选择题答案

1．B　　2．D　　3．B　　4．A　　5．D

二、填空题答案

1．require()语句会输出错误信息，并且立即终止脚本的处理，而 include()语句在没有找到文件时会输出警告，不会终止脚本的处理、require_once 或 include_once

2．面向对象程序设计、面向过程程序设计

3．sum=15

4．sum=55

5．会看到

第 5 章　PHP 数组

一、选择题答案

1．B　　2．A　　3．C　　4．D　　5．D

二、填空题答案

1．size

2．array_push

3．array_unique()

4．数字索引数组

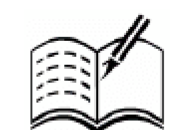
5．一维数组、二维数组、多维数组

第 6 章　Web 技术

一、选择题答案

1．D　2．C　3．A　4．D　5．B　6．B　7．A　8．A　9．C　10．C





第 7 章　MySQL 数据库

一、选择题答案

1．D　　2．D　　3．A　　4．D　　5．A

二、填空题答案

1．delete from tb_user where id=5
2．mysql_query("set names utf8");
3．MYSQLDUMP 命令、MYSQL 命令
4．select * from tb_mrbook order by price desc limit 3;
5．0、1、2、3、4、5 条记录的所有字段信息

第 8 章　PHP 数据库编程技术

一、选择题答案

1．A　　2．D　　3．A　　4．B　　5．A

二、填空题答案

1．name='".$_POST['name']."'
2．
$conn=　mysql_connect　("localhost","root","111");
mysql_select_db　("db_database08",　$conn　);
mysql_query　("set names utf8");
3．mysql_fetch_array()和 mysql_fetch_row()
4．id<5
5．'mr','123456'

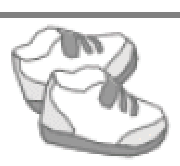
第 9 章　字符串高级处理

一、选择题答案

1．A　　2．A　　3．B　　4．A　　5．B

二、填空题答案

1．Explode,implode,print



2．This,course,is,very,easy!
3．POSIX　PCRE
4．strlen
5．preg_match



第 10 章　日期和时间处理

一、选择题答案

1．C　　2．B　　3．C　　4．C　　5．B

二、填空题答案

1．Strtotime("1 January 2100")
2．Date_default_timezone_set("Asia/Shanghai");
3．mktime(date("Y-m-d"));
4．

```
date_default_timezone_set("Asia/Shanghai");
    $str=mktime("2010-12-10");
    $str=getdate($str);
    echo $str[weekday];
```

5．今天不是一个特殊的日子！

第 11 章　图形图像处理

一、选择题答案

1．C　　2．B　　3．A　　4．C　　5．C

二、填空题答案

1．$image=imagecreate(800,800);
2．imagedestroy;
3．imagecreate;
4．imagestring;
5．130,30

第 12 章　文件、目录处理

一、选择题答案

1．C　　2．C　　3．A　　4．B　　5．C



Note

二、填空题答案

1. $array=$_FILES['file']
2. rmdir
3. Move_uploaded_file
4. Is_uploaded_file
5. 下载

第 13 章 面向对象编程

一、选择题答案

1. C　　2. A　　3. C　　4. C　　5. B

二、填空题答案

1. Abstruct class People{}
2. clone
3. static
4. instanceof
5. 我是初始化函数，在实例化的同时就被调用了

第 14 章 PDO 数据库抽象层

一、选择题答案

1. B　　2. A　　3. A　　4. C　　5. B

二、填空题答案

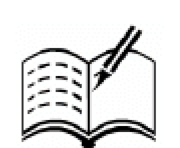
1. PDO::ERRMODE_SILENT
2. PDO::ERRMODE_WARNING
3. PDO::ERRMODE_EXCEPTION
4. Exec
5. PDO 连接 MySQL 成功

第 15 章 Smarty 模板

一、选择题答案

1. A　　2. A　　3. D　　4. D　　5. D





二、填空题答案

1. templates
2. $smarty->caching=true
3. display
4. assign
5. $smarty->assign("value",$rs);

第 16 章　ThinkPHP 框架

一、选择题答案

1. A 、B　　2. B　　3. A　　4. A、B、C、D　　5. A、B、C、D

二、填空题答案

1. Conf 目录
2. Lib\Action
3. 创建、更新、读取和删除
4. .html，TMPL_TEMPLATE_SUFFIX
5. assign 方法

第 17 章　PHP 的字符编码

一、选择题答案

1. A　　2. B　　3. C　　4. A　　5. C

二、填空题答案

1. <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
2. mb_detect_encoding()
3. header("Content-type:text/html;charset=utf-8");
4. mb_substr()
5. iconv("utf-8","gbk","PHP");